**Trabajo de Fin de Grado**
**Grado en Ingeniería Informática**

# Desarrollo de un sistema multimodal de simulación de entrevistas de trabajo para dispositivos móviles


**Autor**: Luis Buceta Ojeda
**Tutor**: David Griol Barres

2018

# Índice general

# Índice de figuras

# Índice de tablas

# Capítulo 1

# Introducción

En este primer apartado, se procederá a describir, brevemente, el presente Trabajo de Fin de Grado. En los siguientes subapartados, se destacará la motivación del proyecto, los objetivos de este, y la estructura del resto de este documento.

## 1.1. Motivación

El procesamiento del lenguaje natural es un campo de la computación (en concreto, con raíces en la inteligencia artificial) basado en la interacción entre el lenguaje utilizado por los humanos en su día a día y el utilizado por el mundo informático. El objetivo principal de este campo científico es estudiar y desarrollar sistemas que permitan a las personas utilizar el lenguaje natural para interactuar con aplicaciones o sistemas informáticos. Las ventajas que tienen los sistemas con capacidades de procesamiento del lenguaje son manifiestas, pero hay una que destaca por encima de cualquier otra: la capacidad de acercar al usuario al sistema, presentando a este una experiencia más reconocible y familiar, y que le permite utilizar su propia experiencia pasada para realizar las actividades propuestas por el sistema. Aunque en la Era de la Información, donde el uso de ordenadores encontrados en múltiples formatos y estilos (desde teléfonos móviles a ordenadores portátiles) está a la orden del día y donde el nivel de habilidad con estos aumenta exponencialmente, la utilización del lenguaje natural ofrecería un amplio abanico de oportunidades a muchos usuarios que desean utilizar estos aparatos con mayor destreza, pero que se ven obstaculizados por o bien una falta de conocimiento de los procedimientos necesarios para hacer que el sistema funcione, o bien por impedimentos de salud (por ejemplo, problemas de vista que impiden ver a un usuario la pantalla y, por ello, interactuar con ella).

Es por estas razones que el procesamiento del lenguaje natural es uno de los pilares de la llamada interacción multimodal [1]. La interacción multimodal, dentro del campo de las ciencias de la informática, se define como la existencia de múltiples vías de interacción sobre un sistema, permitiendo que un usuario utilice distintas herramientas de entrada y salida de información. Por ello, se puede decir que un sistema multimodal es aquel que interactúa con los usuarios mediante múltiples canales, como puede ser la utilización del audio, el video o el texto, y no utilizando solo un método de interacción [2]. La interacción multimodal entre una máquina y un humano busca que la experiencia del usuario sea flexible como puede ser una experiencia con otro humano,

y así conseguir una usabilidad y accesibilidad del sistema todavía mayor. Dentro de la interacción multimodal, tal y como se explicaba anteriormente, se suelen combinar distintas herramientas de entrada al mismo tiempo, como pueden ser un teclado junto a un detector del habla, o la utilización de vídeo para reconocer gestos o movimientos. De la misma manera, los sistemas ofrecen varias herramientas de salida, como pueden ser motores de síntesis de texto a voz *Text-To-Speech* (TTS) o la utilización de imágenes o vídeos para representar cierta información, además de, por supuesto, texto, que es el canal más común.

Los sistemas multimodales traen consigo un alto número de ventajas relativas a la accesibilidad de uso por parte de los usuarios, pero también pueden conseguir que el usuario interaccione con el sistema de una manera diferente a los canales habituales con los que se suele interactuar con programas informáticos (normalmente, utilizando elementos como teclados, ratones o superficies táctiles) y que eso conlleve que el sistema pueda realizar análisis alternativos a la entrada del usuario. La capacidad de utilizar herramientas de reconocimiento del habla, como pueden ser motores de *Speech-To-Text* (STT), ofrecen la capacidad de utilizar análisis de minería de texto ante respuestas del usuario realizadas en un entorno más natural que el utilizado escribiendo un respuesta por teclado, dada la familiaridad del usuario con el habla natural, lo que ofrece resultados más cercanos a los perceptibles en una conversación entre dos personas [3]. Con la posibilidad de realizar análisis más certeros que los que se tendrían en un sistema basado en la escritura por teclado, se ofrecen oportunidades interesantes para estudiar las respuestas naturales de los usuarios ante simulaciones de situaciones concretas, y se podría hacer uso de estos análisis para enriquecer el conocimiento recogido en dichas simulaciones para ofrecérselo al usuario como sistema de retroalimentación que pueda ayudarle a mejorar cuando, potencialmente, se encuentre personalmente en situaciones similares.

Una de estas potenciales situaciones es conocida por más de 22 millones de españoles que componen la población activa del país durante el primer trimestre de 2018 [4]. Las entrevistas de trabajo son reuniones formales, realizadas entre un candidato a un puesto de trabajo y uno o más responsables de reclutamiento de la empresa implicada, con el objetivo de cuestionar distintos aspectos del candidato para comprobar su idoneidad para el puesto. Estas entrevistas son determinantes a la hora de decidir nuevas contrataciones dentro de las empresas [5], y de cara al candidato su resultado es muy obtuso. La empresa decide unilateralmente como interpretar las respuestas de los candidatos y, de ahí, sacar conclusiones relativas a estos, y los candidatos solo acaban recibiendo una respuesta binaria acerca de su resultado (bueno o malo). Los entrevistadores no ofrecen retroalimentación del rendimiento en la entrevista a los candidatos, al igual que no revelan a estos su sistema de evaluación. Teniendo en cuenta esto, más los avances tecnológicos relatados anteriormente, y el uso potencial del habla natural ante un sistema computacional como puede ser un móvil, se puede pensar en la simulación de entrevistas de trabajo como herramienta de retroalimentación para desenvolverse en situaciones similares como un campo en el que un cierto marco de usuarios puede estar interesado, puesto que, un sistema que pudiese ser capaz de simular con precisión una entrevista laboral, podría ofrecer a sus usuarios la capacidad de poder practicar este proceso, hacerles conocer sus virtudes y defectos y, así, ayudarles a enfrentarse a

entrevistas reales en el futuro.

## 1.2. Objetivos

Teniendo en cuenta lo comentado en el apartado anterior, este Trabajo de Fin de Grado tiene como objetivo principal el desarrollo de un sistema de simulación de entrevistas de trabajo que, a través de una interacción multimodal basada en el habla y el texto, será capaz de ofrecer a los usuarios un análisis de su rendimiento en dichas entrevistas.

El usuario interactuará, mediante el uso del habla o directamente escribiendo con teclado, con un agente conversacional que realizará dos papeles:

- Mantendrá el flujo de conversación entre el usuario y el propio agente, realizando preguntas establecidas en la entrevista, y recogiendo la respuesta del usuario.
- Estudiar de manera semántica la respuesta del usuario, y hacer un análisis de la relación de esta con el contexto de la pregunta planteada

Este agente conversacional, dentro del concepto de la simulación, hará el papel del entrevistador encontrado en una entrevista real, y sus reglas de conversación deberán estar definidas previamente con la idea de realizar la simulación más adecuada posible, tanto en tipo de preguntas realizadas, como en el análisis contextual realizado posteriormente.

Paralelamente, el sistema utilizará un sistema de análisis, basado en la minería de texto y el análisis de sentimiento textual, que estudiará el tono y sentimiento de la respuesta del usuario, para obtener una representación de la intención del usuario al generar dicha respuesta. Este análisis, junto con el realizado por el agente conversacional, será ponderado por el sistema y mostrado al usuario como evaluación de su participación en la simulación, para que este pueda interpretar sus resultados adecuadamente y pueda mejorar a la hora de responder ciertas preguntas de cara a entrevistas de trabajo reales.

Por último, concretar que, aunque la interacción del usuario con este sistema podría estar definido en aplicaciones disponibles en múltiples plataformas (que podrían realizar funcionalidades similares a la aplicación presentada en este proyecto), este Trabajo de Fin de Grado define la aplicación de interacción dentro de un entorno Android, sistema operativo presente en gran mayoría de los dispositivos móviles disponibles actualmente, y que permite que una gran cantidad de usuarios puedan utilizar este sistema en este tipo de dispositivos.

## 1.3. Estructura de la memoria

Este documento se compone de cinco secciones principales, cada una con un objetivo concreto con la intención de ofrecer contexto y explicación del desarrollo de este

Trabajo de Fin de Grado. Además, la bibliografía encontrada al final del documento destaca todas las referencias utilizadas en el desarrollo del proyecto. Estas cinco secciones principales son:

- **Introducción**. Esta misma sección ofrece una explicación básica de la intención de este proyecto, los objetivos encontrados en él, y esta misma sección que explica la estructura de la memoria.
- **Estado del arte**. Un análisis de las tecnologías y conceptos relacionados con este proyecto utilizados en la actualidad, y que ayudan a definir el contexto del proyecto y la razón de su existencia.
- **Diseño y desarrollo de la aplicación**. En esta sección se definirán tres elementos del procedimiento seguido para realizar este trabajo: el diseño de la aplicación, su desarrollo y el proceso de pruebas seguido para comprobar su funcionamiento.
- **Gestión del proyecto**. Aquí se detallarán los recursos necesarios para realizar este proyecto, y un desglose temporal del desarrollo del proyecto, además de un análisis del marco regulador en el que se encuentra y el entorno socioeconómico del proyecto.
- **Conclusiones**. Esta última sección ofrece un resumen de los conocimientos adquiridos en la realización de este proyecto, además de futuras mejoras potenciales que podrían ser aplicadas.

# Capítulo 2

# Estado del arte

Este capitulo tiene como finalidad detallar el contexto teórico y tecnológico en el que se encuentra este trabajo. En él, se detallarán múltiples conceptos utilizados por el sistema de simulación, empezando por una definición de los sistemas conversacionales o de diálogo hablado, tanto dentro como fuera de un entorno Android como el utilizado en el trabajo, que proponen la base de la interacción entre el usuario y el sistema de simulación de entrevistas. Además del concepto, se concretará y definirá la interfaz de diálogo utilizada para ofrecer esta funcionalidad a los usuarios, llamada *Dialogflow* [6], que es capaz de, tal y como se definió en los objetivos, mantener un flujo de conversación con los usuarios y realizar un análisis semántico de sus respuestas.

Además, se definirá el concepto de Análisis de Sentimientos (*Sentiment Analysis*), una pieza importante del procesamiento de respuestas del usuario dentro de la aplicación, y que se encarga de ofrecer un estudio de dichas respuestas y de ello extrapolar un sentimiento o tono con el que dicha respuesta ha sido formulada. De cara al sistema de simulación, este análisis ofrece la posibilidad de comprobar la intención básica del usuario a la hora de responder, y tener una imagen más completa y realista para realizar la simulación. Este análisis de sentimientos está integrado en la aplicación mediante la conexión a varios servicios especializados en este tipo de analíticas, que son capaces de recibir peticiones mediante su API y devolver un análisis de sentimientos completo en segundos. En este capítulo también se explicará el funcionamiento y capacidades de estos, concretando qué servicios han sido utilizados en este trabajo para realizar un análisis de sentimientos preciso y completo.

Por último, se explicará brevemente conceptos relacionados con entrevistas de trabajo y su estructura, la cual suele ser definida antes de su realización por parte de los entrevistadores teniendo como referencia ciertos patrones y estructuras básicas, y que buscan revelar detalles concretos de los candidatos antes de realizar la decisión de la contratación.

## 2.1. Sistemas de diálogo hablado

La definición primordial de un sistema de diálogo hablado (*spoken dialog system* o sistemas conversacionales) es que es un sistema informático que recibe, por parte de

un usuario, una entrada basada en frases realizadas con lenguaje natural y de carácter oral, y que devuelve una salida con frases generadas, de nuevo, con lenguaje natural y en formato oral [7]. En un nivel abstracto, esta definición es aplicable también a una conversación oral entre dos humanos, y es que los sistemas de diálogo hablado buscan que la interacción entre un usuario y un sistema informático sea lo más parecida posible a la que tendría con un otra persona física.

Esta comparación con la interacción humana no se detiene en la definición, y es que los sistemas de diálogo hablado están compuestos por módulos que, conceptualmente, realizan un ciclo de ejecución comparable al realizado por una persona que recibe nueva información, y que debe responder con criterio a esta. Para ello, hace uso de varios módulos internos, que resultan en un flujo computacional que acaban ofreciendo al usuario del sistema una respuesta dentro del contexto de las frases de entrada y en formato hablado.

En la Figura 2.1, se puede ver un esquema de este flujo, y como el sistema utiliza cada uno de sus módulos para acabar generando esa respuesta final que presentará al usuario. El procedimiento básico se define como la entrada de un mensaje por parte del usuario al sistema, un proceso de comprensión por parte de este, una generación de una respuesta y la reproducción de dicha respuesta al usuario.

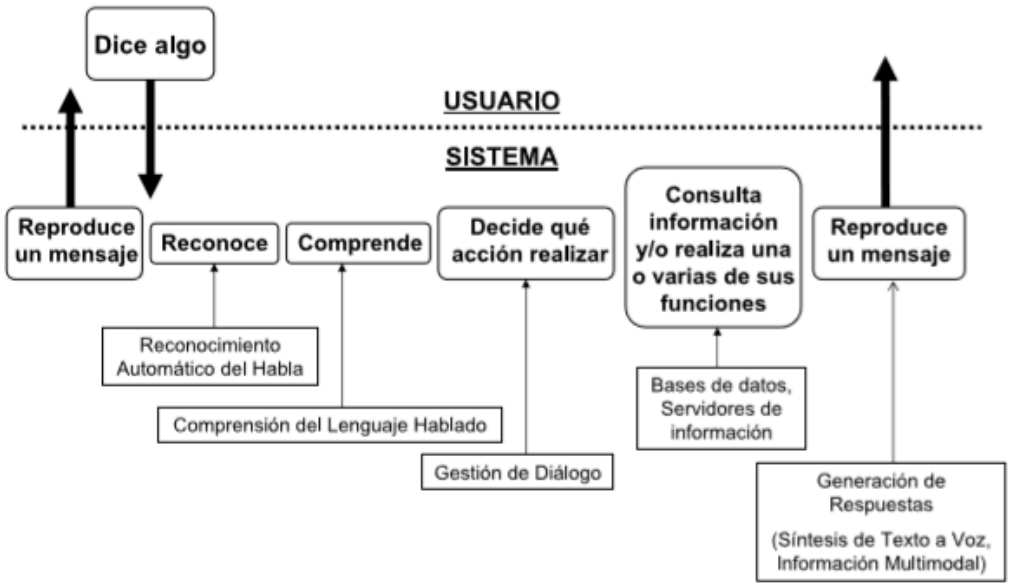


Figura 2.1: Diagrama de acciones de un sistema de diálogo hablado [8]

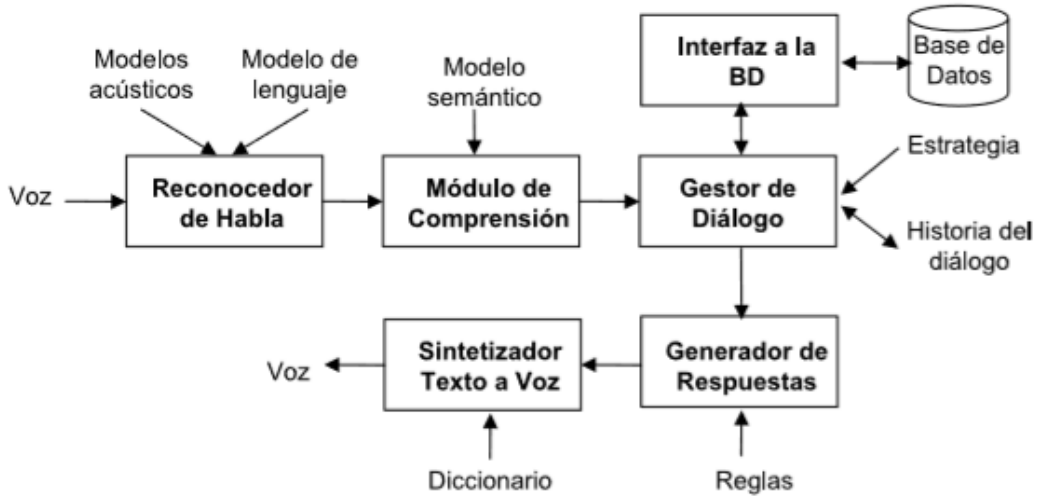


Figura 2.2: Diagrama modular de un sistema de diálogo hablado [8]

Como se puede apreciar, hay cinco módulos que definen un sistema conversacional y que le otorgan la capacidad de computación necesaria para la generación de respuestas finales. En la anterior Figura 2.2, se puede ver una estructura más detallada de estos módulos y la interconexión entre ellos. Los módulos son los siguientes:

- **Sistema de reconocimiento automático del habla**, o también conocido como ***Speech-To-Text (STT)***. Este sistema reconoce los sonidos emitidos por el habla del usuario, y extrae la información contenida en este habla para transformarla a texto que pueda ser estudiado en posteriores módulos.
- **Comprensión del lenguaje hablado**. Este módulo estudia el texto recibido y realiza un análisis semántico de él, el cual utiliza para recoger los términos y elementos clave de la entrada y así poder definir la respuesta de una manera más precisa.
- **Gestión del diálogo**. Utilizando el análisis de comprensión e información extra obtenida del siguiente módulo (consultas en bases de datos y servidores externos), este módulo se encarga de generar la nueva respuesta en texto que será, en el último módulo, transmitida al usuario.
- **Módulo de consulta de información**. Este módulo, a partir del análisis semántico anterior, realiza consultas a elementos externos al sistema, como pueden ser bases de datos o servidores de información externos, o el historial de, que complementan el conocimiento recibido y que ayudan al módulo de gestión a elegir una respuesta final que mostrar.
- **Generación de respuestas**, o también conocido como ***Text-To-Speech (TTS)***. El último módulo se encarga de enviar al usuario la respuesta final generada por el gestor de diálogo, sintetizando el texto generado a voz, la cual se reproduce al usuario.

Con todos estos módulos, se puede observar que el flujo de ejecución es similar al que realizaría una persona de cara a responder a otra. Esta persona recibiría la información por el oído, pensaría en que significa esta nueva información, recogiendo los elementos clave de la frase, y generaría una nueva respuesta que expresaría por voz al otro miembro de la conversación. Este acercamiento a la naturaleza humana por parte de un sistema informático significa una mayor accesibilidad de cara a los usuarios que quieran interactuar con la aplicación, lo que también es uno de los pilares de la interacción multimodal.

### 2.1.1. Sistemas conversacionales con interacción multimodal

Tal y como se definió en la introducción, la interacción multimodal se define como la existencia de múltiples vías de interacción sobre un sistema, permitiendo que un usuario utilice distintas herramientas de entrada y salida de información. Es por ello que un sistema multimodal es el que permite que los usuarios interactúen por varios canales (audio, video, texto), y no únicamente por un único canal, para realizar una tarea concreta [2].

La primera demostración de sistema multimodal surgió en 1980, en el *MIT Architecture Machine Group*, en un proyecto dirigido por Richard Bolt llamado "***Put That There***". Proyectado sobre una pared, el sistema ofrecía la capacidad de utilizar comandos de voz y gestuales para dibujar objetos espaciales sobre la superficie, utilizando definiciones descritas por voz (como, por ejemplo, "dibuja un cuadrado azul ahí") con gestos apuntando a la posición donde se quería dicho objeto. A esta implementación le siguieron otros projectos que tenian como objetivo la implementación de más opciones de interacción para que los usuarios de los sistemas pudiesen realizar tareas más complejas utilizando sus conocimientos naturales y experiencias previas sin necesidad de haber utilizado el sistema anteriormente. Ejemplos de estos sistemas originales son *CUBRICON* en 1989 (que utilizaba varios tipos de introducción de lenguaje natural junto con gráficos y mapas para simular el planteamiento táctico de misiones) o *QuickSet* en 1997 (un sistema de entrenamiento de Marines de EEUU que utilizaba tanto interacción táctil como verbal) [1].

Otro sistema destacable, surgido en el año 2002, es el sistema **MATCH** (siglas para *Multimodal Access To City Help*). En 2002, para los usuarios medios, los dispositivos portátiles estaban limitados a ordenadores portátiles, teléfonos móviles con funcionalidades limitadas en comparación a los *smartphones* actuales y PDAs, y ninguno de estos tenía capacidades multimodales desarrolladas. Actualmente, la utilización de mapas online, como pueden ser Google Maps o Apple Maps (ambos disponibles en una multitud de dispositivos), ayudan a que los usuarios puedan orientarse en una población, y a realizar búsquedas de calles, tiendas o restaurantes instantaneamente. Esta interacción con estos servicios de mapas puede realizarse de múltiples maneras (escribiendo una dirección, mediante servicios de geolocalización o por voz), y el proyecto MATCH consiguió en 2002 un primer modelo de interacción multimodal para interactuar de estas maneras con un servicio de mapas.

Utilizando tanto interacción hablada, como escrita y táctil, los usuarios de MATCH eran capaces de preguntar al sistema por localizaciones genéricas (por ejemplo, restaurantes italianos) dentro de una localización, y mediante un sistema conversacional integrado, este era capaz de preguntar más datos al usuario para acotar la búsqueda. Los usuarios podían seguir conversando con el sistema hasta llegar al resultado deseado, o bien utilizar un lápiz táctil para desplazarse por el mapa y hacer zoom, y poder cotejar los resultados ellos mismo [9]. Como se puede ver, este sistema MATCH y el resto de ejemplos nombrados realizan funciones similares a las disponibles en dispositivos actuales, pero la necesidad de utilizar hardware dedicado para conseguir una correcta interacción multimodal evitó que su expansión fuese amplia de cara a los usuarios. Los sistemas multimodales fueron avanzando con la tecnología que los sostenía, y la aparición de sistemas móviles, como *smartphones* o *tablets*, cuyo hardware incluyen por defecto un alto número de sistemas de interacción (pantallas táctiles, micrófonos, acelerómetros...) ofrecen la posibilidad de que sistemas con modelos de interacción multimodal estén al alcance de un alto número de usuarios, y con ello todas las ventajas que ofrecen y que, en el pasado, no eran accesibles para el usuario medio con los dispositivos disponibles [1]. En la Figura 2.3 se puede ver el sistema MATCH, que estaba integrado dentro de una tableta con capacidades físicas similares a las tabletas actuales.

Figura 2.3: Muestra de uso del sistema MATCH [9]

Los sistemas multimodales ofrecen un número de ventajas interesantes respecto a sistemas unimodales. El objetivo principal de los sistemas multimodales es ofrecer a los usuarios la posibilidad de interactuar de manera más natural y eficiente con sistemas informáticos complejos, y, en comparación a sistemas unimodales, la interacción multimodal cumple este objetivo considerablemente mejor, según estudios realizados con niños [10] y adultos [11]. Las ventajas observadas durante la evolución de los sistemas multimodales se basan en el uso flexible de varios sistemas de introducción de datos, lo

que aumenta la eficiencia del usuario a la hora de encontrarse con sistemas con los que nunca ha interactuado antes. Los usuarios, al poder interactuar al mismo tiempo con sistemas táctiles, de habla, escritos o gestuales, son capaces de superar las limitaciones intrínsecas de cada uno de esos sistemas por separado, y lograr una interacción basada en las experiencias pasadas y naturales que resulta en una experiencia más adecuada y eficiente [1], en general la experiencia de uso resulta ser mejor. Además de esto, la interacción multimodal tiene una base de accesibilidad. Los usuarios, al poder interactuar con un sistema de distintas formas, y no limitándose a una sola forma que no puedan utilizar (por ejemplo, la discapacidad de leer texto por tener discapacidades visuales o la de escuchar sonidos por problemas auditivos), pueden llegar a utilizar los sistemas de la misma manera que lo haría cualquier otra persona.

Estos beneficios son aparentes, pero la interacción multimodal tiene otra ventaja sobre otros sistemas de interacción regular, y es la capacidad de utilizar estos métodos alternativos de transmitir información para configurar el sistema de manera que el servicio ofrecido utilice aspectos alternativos de la información recibida. Un ejemplo claro, de la misma manera que utilizaron en el proyecto de *"Put It There"*, sería la colocación de objetos en un espacio. En un sistema unimodal, esta interacción podría realizarse con más o menos éxito (potencialmente la posiblidad de dictar a un sistema las coordenadas donde se quiere un objeto funcionaría), pero el esfuerzo necesario por el usuario implica dos puntos importantes a la hora de diseñar el sistema [1]:

- **Reducción del interés del usuario**. Dependiendo del marco de usuarios esperados por el sistema, es posible encontrar situaciones en donde la falta de una interacción cómoda para el usuario implique la reducción de uso del sistema.
- **Los datos pueden ser erroneos**. El usuario puede no tener los conocimientos necesarios para realizar la actividad planteada con los sistemas de *input* encontrados (por ejemplo, saber qué coordenadas son necesarias para colocar un objeto en un espacio). Esto implica que el análisis de datos deje de ser fiable, y que el tanto el sistema como el usuario tengan problemas en realizar la tarea correspondiente.

El segundo punto es importante para comprender la integración de los sistemas multimodales en un sistema de diálogo hablado. El habla se puede definir como la principal vía de comunicación entre humanos, utilizando la voz, pero no es el único método de comunicación que existe entre personas. La capacidad de escribir y leer, o de utilizar lenguajes alternativos no basados en el sonido, como pueden ser lenguajes de signos, permiten una comunicación a través de canales alternativos. Sin embargo, el habla es el canal más entrenado por la gente, ya que se comienza a utilizar desde una edad temprana. Es por ello que la capacidad de conversar es una de las más utilizadas por las personas, y posiblemente la que resulta más sencilla para la mayoría de los humanos [12].

Un sistema conversacional puede tener entradas y salidas de distintos tipos, siempre y cuando sean expresables con lenguaje natural. Es por ello que, para utilizar sistemas de diálogo hablado, se utilizan entradas de habla, que son transformadas en texto para un correcto uso interno del fragmento introducido por parte de los módulos

correspondientes del sistema. Sin embargo, la capacidad de utilizar sistemas de interacción multimodal ofrecen más posibilidades a los usuarios para interactuar con los sistemas conversacionales [13]:

- **Habla**. El método principal de interactuar con un sistema de conversación hablado es el habla. Ofrece al usuario la capacidad de expresarse de forma natural y conocida, lo que hace que la introducción de información en el sistema sea más cercano al encontrado en una conversación real. Para ello, se pueden utilizar módulos de reconocimiento del habla (también llamados *Speech-To-Text*), que convierten el habla natural del usuario en texto plano que puede ser utilizado por el sistema. Tener una entrada de información que utiliza el principal canal de comunicación entre los humanos ofrece la posibilidad de ajustar el sistema conversacional para ofrecer un estudio interno más preciso y una generación de respuestas más realista y consistente con la entrada recibida.

- **Texto escrito**. Saltándose un paso de proceso (el procesamiento de voz por parte del sistema y convertirlo a texto), el usuario es capaz de escribir la información pertinente en vez de expresarla vocalmente. Aunque potencialmente el usuario podría escribir el mismo texto que expresaría hablando, la longitud de los textos escritos son más cortos que los hablados, y suelen llevar el mínimo de información posible, especialmente escritos en dispositivos móviles [14], lo que conlleva que la información recibida pueda no ser tan precisa como la recibida mediante comunicación hablada, y que eso resulte en un rendimiento del análisis por parte del sistema de diálogo hablado inferior al esperado.

- **Gestos**. Utilizando sistemas de reconocimientos de gestos, sería posible recoger una conversación basada en lenguajes de signos, que podría ser interpretada por parte del sistema de la misma manera que una entrada hablada. En investigaciones se ha comprobado que el ratio de acierto puede llegar a ser muy inestable: sin utilizar ningún tipo de hardware especial y con símbolos estáticos, el índice de acierto puede estar entre un 39 % y un 99 %, mientras que con hardware especializado y símbolos dinámicos puede estar entre el 61.3 % y un 97.4 % de acierto [15]. Ese índice de acierto considerablemente errático puede dar problemas a la hora de reconocer fragmentos de información largos, e implicaría un ajuste grande del sistema conversacional para poder trabajar correctamente con datos de este estilo de manera que el estudio sea preciso y las respuestas generadas sean comprensibles.

Por su parte, la salida generada por el sistema también entra dentro del concepto multimodal, y es que la información generada por el sistema de diálogo hablado es capaz de devolver una respuesta a los usuarios de distintas maneras, y, de la misma manera que la entrada del usuario, esta capacidad permite ofrecer más posibilidades a los usuarios para interpretar el proceso generado por el sistema. Algunas posibilidades encontradas para devolver respuestas al usuario son las siguientes:

- **Habla**. Mediante módulos de síntesis de voz, o también llamados *Text-To-Speech*, los sistemas conversacionales son capaces de utilizar una fuente de texto como base para generar archivos de audio con voz sintetizada digitalmente, que replican la capacidad de habla de los humanos y que ofrecen al usuario una representación reconocible a la que utiliza diariamente en un entorno real con otras personas.

- **Texto**. Internamente, los sistemas de diálogo hablado utilizan texto plano para realizar los análisis pertinentes para poder generar una respuesta. Es por ello que una salida por texto es la salida más básica que puede generar un sistema conversacional, ya que representa la respuesta generada sin modificar y sin cambios de formato. Aunque leer una respuesta puede ser más rápido que escucharla, los humanos tienden a leer con velocidad y saltandose palabras y conceptos, que dependiendo de la complejidad de los textos, puede llevar a problemas de comprensión [16]

- **Representaciones virtuales humanas**. Siendo un sistema informático, el sistema de diálogo hablado es capaz de comunicarse con otros sistemas alternativos que puedan proporcionar funcionalidad adicional a la realizada dentro del agente conversacional. Un sistema con el que, potencialmente, podrían comunicarse y utilizar como apoyo para transmitir información es el sistema de un agente tridimensional que represente a una figura humana, representada por pantalla, que pueda reproducir habla sintetizada y emociones simuladas. Teniendo una figura reconocible, los usuarios tendrán más afinidad con el sistema y la interacción con ellos será más natural, al ser más cercana a poder interactuar con otro humano [17]. En la Figura 2.4, se puede ver un ejemplo de una integración de una representación humana virtual en un sistema conversacional, que es capaz de mostrar diferentes emociones de cara a que el usuario empatice más con el agente.

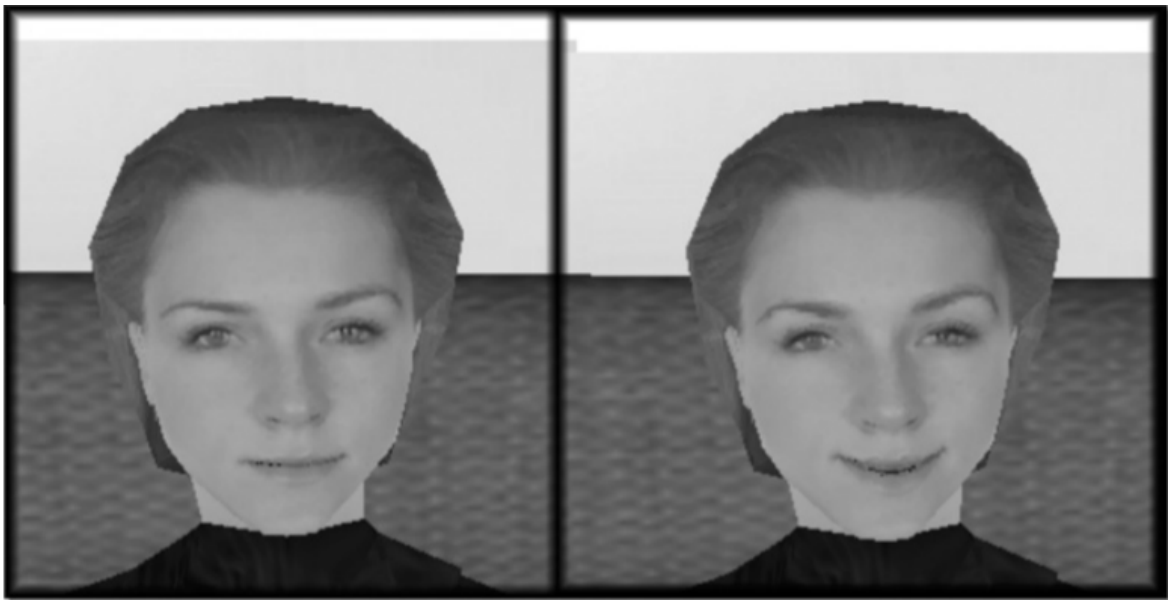

Figura 2.4: Agente virtual con expresión neutra (izq.) y expresión sonriente (der.) [18]

### 2.1.2. Sistemas de diálogo hablado en el entorno Android

El sistema de simulación de entrevistas de trabajo que se define en este Trabajo de Fin de Grado utiliza como plataforma principal el sistema operativo Android, presente en múltiples dispositivos móviles, como teléfonos móviles, relojes inteligentes y hasta televisiones.

Android es un sistema operativo basado en Unix, y cuyo principal enfoque se basa en la interacción con dispositivos móviles mediante, principalmente, sistemas táctiles, pero también sonoros y gestuales. Nacida en 2003, y con intención de crear un sistema operativo para cámaras digitales, Android Inc. fue adquirida por Google en 2005 [19]. En 2008, la primera versión de su sistema operativo, Android, fue lanzada al consumidor con funciones limitadas en comparación a versiones posteriores de Android que, en un ciclo anual, han ido aumentando y puliendo las funcionalidades del sistema operativo tanto para usuarios como para desarrolladores de aplicaciones.

Gracias a su código fuente abierto, y la predisposición por parte de Google de ofrecer facilidades a los desarrolladores tanto para crear aplicaciones de múltiples utilidades y funcionalidades (y ofrecerlas dentro de una tienda integrada en el sistema operativo que ofrece la posibilidad de descargar estas aplicaciones en cualquier dispositivo disponible) como para mejorar el sistema operativo y ofrecer soporte extraoficial a dispositivos antiguos (cuyo soporte oficial por parte de las compañias es reducido), Android se convirtió en el sistema operativo móvil más utilizado globalmente, con un índice de usuarios activos cercano al 85 % durante el 2017, en comparación al 14.7 % del iOS de Apple, o el 0.1 % de Windows Phone (de Microsoft), tal y como se puede ver en las Figuras 2.5 y 2.6.

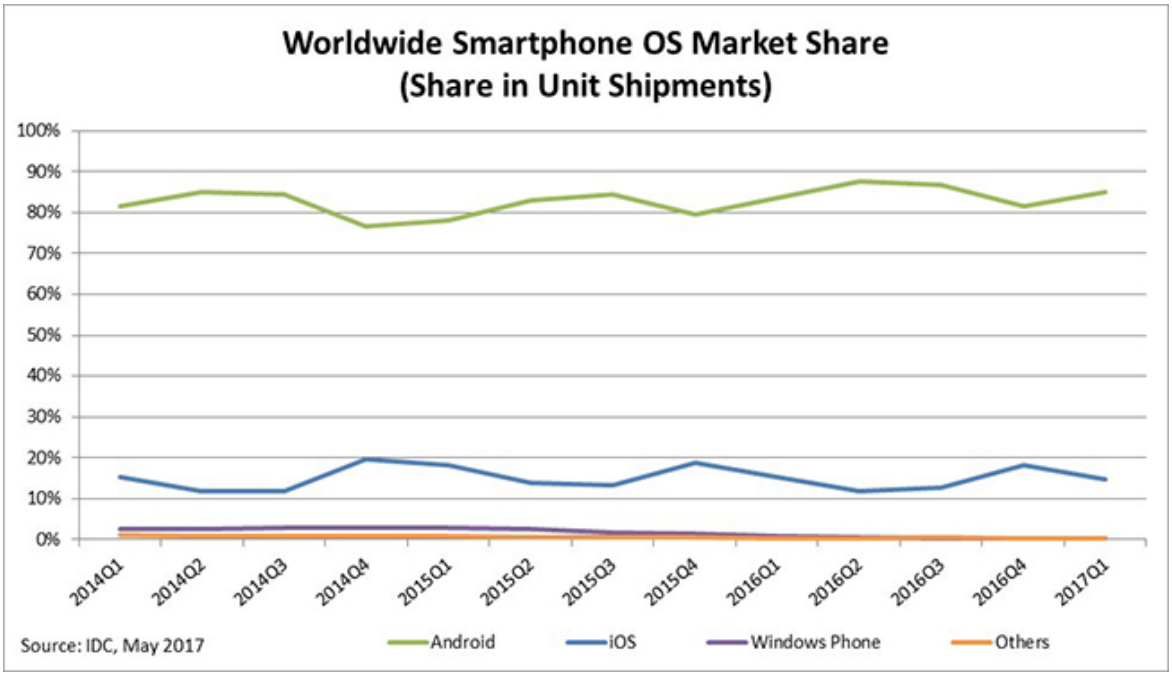


Figura 2.5: Gráfico de uso de sistemas operativos móviles (de 2014 a 2017) [20]

| Period | Android | iOS | Windows Phone | Others |
|---|---|---|---|---|
| **2016Q1** | 83.4% | 15.4% | 0.8% | 0.4% |
| **2016Q2** | 87.6% | 11.7% | 0.4% | 0.3% |
| **2016Q3** | 86.8% | 12.5% | 0.3% | 0.4% |
| **2016Q4** | 81.4% | 18.2% | 0.2% | 0.2% |
| **2017Q1** | 85.0% | 14.7% | 0.1% | 0.1% |

Figura 2.6: Desglose de porcentajes de uso de sistemas operativos móviles [20]

Dado que Android es un sistema operativo disponible para una alta multitud de dispositivos y marcas, el soporte es fragmentado. Las actualizaciones de Android son limitadas por los fabricantes del dispositivo, y pueden decidir si adaptar las nuevas versiones de Android a sus dispositivos o no. Es por ello que hay una comunidad externo muy amplia, formada por desarrolladores no profesionales, que realizan este trabajo de soporte extraoficial a dispositivos antiguos o soporte oficial limitado, ofreciendo nuevas versiones de Android y funcionalidad extra a las encontradas en los dispositivos recién salidos de fábrica. Todo esto provoca que el reparto de versiones en los dispositivos Android esté altamente fragmentada, y que no todas las funcionalidades actualizadas del sistema operativo estén disponibles en todos los dispositivos. En la Tabla 2.1 y en la Figura 2.7, se define un desglose de las versiones de Android dentro del conjunto de usuarios del sistema, actualizado a mayo de 2018.

| Versión | Nombre | API | Porcentaje |
|---|---|---|---|
| 2.3.3 - 2.3.7 | Gingerbread | 10 | 0.4 % |
| 4.0.3 - 4.0.4 | Ice Cream Sandwich | 15 | 0.4 % |
| 4.1.x - 4.3 | Jelly Bean | 16-18 | 4.3 % |
| 4.4 | KitKat | 19 | 10.3 % |
| 5.0 - 5.1 | Lollipop | 21-22 | 22.4 % |
| 6.0 | Marshmallow | 23 | 25.5 % |
| 7.0 - 7.1 | Nougat | 24-25 | 31.1 % |
| 8.0 - 8.1 | Oreo | 26-27 | 5.7 % |

Tabla 2.1: Versiones del sistema operativo Android [21]

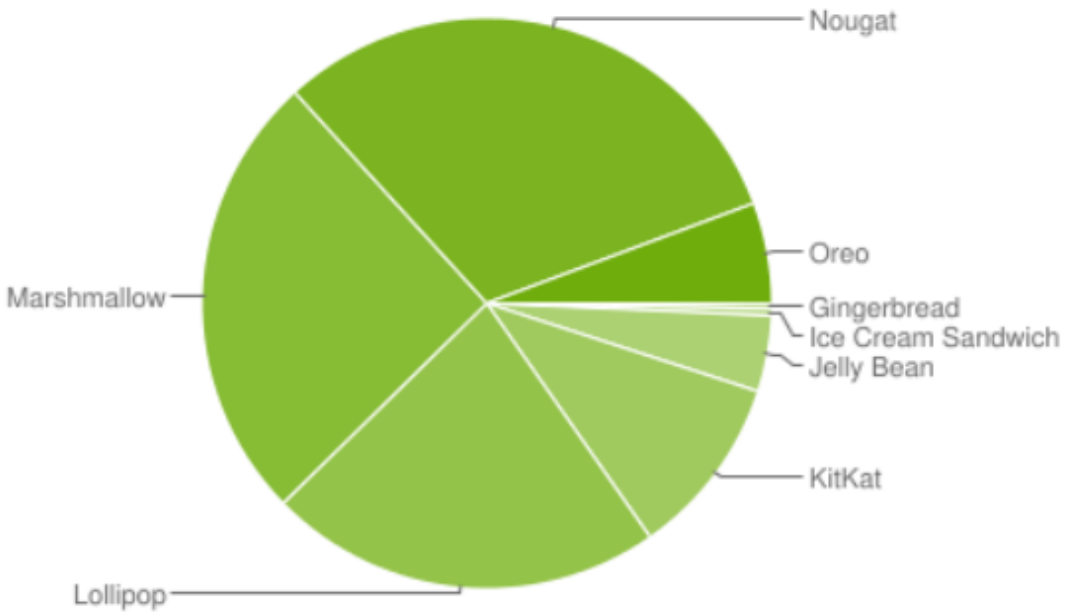


Figura 2.7: Gráfica de despliegue de versiones de Android [21]

Como se puede ver, la distribución de versiones de Android está altamente fragmentada. Versiones como la 4.4 (KitKat), lanzada al público el 31 de octubre de 2013, o inferiores todavía son utilizadas por un 15 % de los usuarios de Android, y más de un 63 % utiliza una versión inferior a los dos últimos lanzamientos (Nougat y Oreo). Esto se debe a lo comentado anteriormente, a un soporte potencialmente limitado por los fabricantes de dispositivos, que dejan ciertos modelos abandonados y sin nuevas actualizaciones. Esto no solo es un problema de cara a utilizar nuevas funcionalidades de Android, si no que también evita que actualizaciones y parches de seguridad sean aplicados en estos dispositivos, lo que pueden generar grandes problemas de privacidad a los usuarios [22].

Una actualización de Android implica también una actualización de la API de desarrollo. Esta API es una interfaz de programación disponible para acceder a elementos y subsistemas del sistema Android, y que ofrecen a los desarrolladores la posibilidad de crear aplicaciones que utilicen las funcionalidades más actualizadas del sistema operativo. Se debe destacar que las aplicaciones creadas en un nivel de API concreto no pueden ejecutarse en versiones de Android que no tengan una versión equivalente o superior de la API objetivo (por ejemplo, una aplicación desarrollada con una API objetivo 19 solo puede ser ejecutada en dispositivos Android que tengan una versión 4.4 KitKat o superior, en versiones inferiores no podrá ejecutarse).

Dada esta limitación por parte de Android para ejecutar aplicaciones de versiones superiores, y la fragmentación de versiones existente actualmente en Android, los desarrolladores no pueden escoger APIs relativamente nuevas para desarrollar sus aplicaciones y, por ello, están limitados en usar las funcionalidades más nuevas de Android

que, potencialmente, podrían ser necesarias para estas, ya que dejarían a un alto número de usuarios sin la posibilidad de utilizarlas y, de esa manera, perder un amplio fragmento del mercado. Esto es relevante a la hora de trabajar con un sistema tan complejo como es un sistema de diálogo hablado. Como se puede encontrar en apartados anteriores, los sistemas de diálogo hablado hacen uso de un alto número de subsistemas y módulos secundarios para realizar la tarea de conversar con los usuarios. Gran parte de estos módulos pueden ser definidos en la parte lógica de la aplicación que sostenga al sistema, pero dos módulos esenciales para el correcto funcionamiento de un sistema conversacional deben utilizar la base ofrecida por el sistema operativo donde el sistema sea ejecutado, que, en el caso de este proyecto, es Android. Estos dos módulos son los que se comunican directamente con el usuario, y recogen la información de este para más tarde compartir la información generada mediante otro canal: el *input* y el *output*.

El ***input*** o introducción de información en la aplicación, como se podía ver en la definición de sistemas multimodales puede realizarse mediante varias maneras. Android, gracias a los dispositivos en los que está presente, tiene una base multimodal: presenta una pantalla donde puede recibir información visual, tanto con imágenes como escribiendo texto, y en la gran mayoría de casos es táctil, lo que permite que los usuarios realicen interacciones con el sistema mediante el tacto. Además, vienen con micrófono y cámara incorporados, lo que deja la posibilidad de que los usuarios utilicen sonidos o gestos visuales para interactuar con el sistema. Sin embargo, siguiendo lo descrito en el apartado anterior, dentro de un sistema de diálogo hablado se destacan las posibilidades presentes del reconocimiento de habla natural, escritura por pantalla o la comunicación gestual, con un enfoque (por definición) en el reconocimiento de habla natural.

- **Reconocimiento de habla**. El reconocimiento del habla natural (el ya mencionado *Speech-To-Text*, o STT) en Android se realiza gracias a módulos integrados en la propia API de Android, en lo que internamente se llama *SpeechRecognizer*. El servicio *SpeechRecognizer* utiliza la entrada del micrófono del dispositivo para escuchar los sonidos externos y reconocer una señal de voz, la cual o bien manda a servidores externos o bien estudia internamente mediante paquetes locales de reconocimiento de voz, y devuelve un fragmento de texto con el mismo contenido que la información hablada. Este servicio entró en la API de Android en su nivel 8 (es decir, en la versión de Android 2.2, también llamada Froyo) [23] [24]. Con este servicio de reconocimiento de habla, Android es capaz de obtener una representación textual de la información hablada por el usuario, y posteriormente utilizar este texto para realizar el análisis interno del sistema de diálogo hablado diseñado. Conociendo la fragmentación de Android, tener un sistema como este en niveles de API reducidos incentiva a los desarrolladores en caso de necesitar una funcionalidad como esta en sus aplicaciones, y en el caso de desarrollo de sistemas de diálogo hablado, este servicio representa la base de la interacción entre el usuario y el sistema.

- **Escritura por teclado**. Además del reconocimiento de habla, Android ofrece la posibilidad de escribir con un teclado virtual integrado en el sistema operativo. Este teclado es equivalente al encontrado en dispositivos como ordenadores personales, y permite al usuario redactar un texto manualmente. Disponible en

el nivel de API 3 (equivalente a la versión de Android 1.5, también llamada Cupcake), es un elemento básico de la interacción con Android para todos sus usuarios, y su disponibilidad en cualquier aplicación es total [25].

- **Interacción gestual mediante cámaras**. Por último, la interacción gestual es también posible en Android, gracias a la utilización de la cámara integrada (de la API 3, pero con una versión actualizada y más precisa en el nivel de API 21, equivalente a la versión de Android 5.0, también llamada Lollipop) [26] [27] y librerías externas como puede ser HandWave, que permite a las aplicaciones usar la cámara frontal para reconocer gestos básicos que podrían ser integrados en el sistema conversacional [28].

En la Figura 2.8, se puede observar la interfaz de Android cuando se accede al módulo integrado de Speech-To-Text desde una aplicación, mostrando un cuadro de información al usuario para que este pueda saber que el sistema operativo le está grabando:

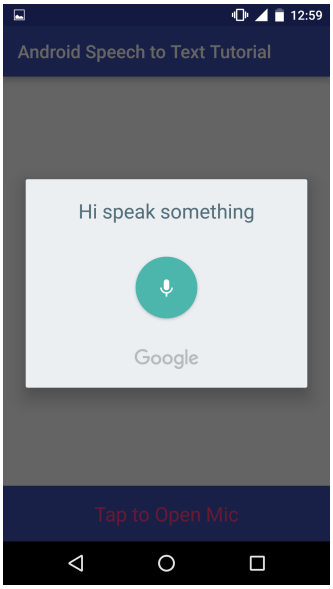


Figura 2.8: Interfaz básica de Speech-To-Text en Android [21]

Por otro lado, el otro elemento de un sistema conversacional que Android puede definir de varias manera es el ***output***, o salida de información. Pudiendo utilizar los mismos recursos que en el *input*, las salidas de un sistema de diálogo hablado pueden ser por voz sintetizada, por texto o con una representación virtual.

- **Síntesis de voz**. Android, al igual que para la entrada de voz, tiene un sistema de generación de voz sintetizada llamado *Text-To-Speech*, o TTS. Este sistema recoge una fuente de texto plano, en este caso la generada internamente por el sistema de diálogo hablado, y genera un archivo de audio digital que contiene una voz sintetizada (encontrada en los paquetes de síntesis de voz incluidos en Android, aunque también descargables desde la tienda integrada) [29] que lee el contenido del texto original, siendo reproducible por la salida de audio del

dispositivo (altavoces, auriculares...). Introducido en el nivel de API 4 (Android 1.4, también llamada Donut), este servicio es uno de los pilares de las opciones de accesibilidad de Android, ya que es el mismo módulo utilizado para leer textos de pantalla para usuarios con problemas de visión que tengan problemas leyendo la pantalla [30]. El bajo nivel de API necesario para utilizar este módulo ayuda a que los desarrolladores lo integren en sus aplicaciones, y dada su similitud al habla humana, puede ser una pieza importante en un sistema de diálogo hablado como el tratado en este proyecto, acercando la interacción del usuario aún más con la encontrada en una conversación humana.

- **Texto**. Por otro lado, gracias a la pantalla encontrada en la mayoría de dispositivos Android, también es posible sacar información por esta, tanto por texto como por imagen. La muestra de texto por pantalla es una de las bases de cualquier sistema operativo, y todo dispositivo Android con pantalla puede hacerlo, siendo el elemento más utilizado por los usuarios de un dispositivo móvil.

- **Agentes virtuales**. Por último, la generación de representaciones humanas virtuales es posible en Android de manera limitada. De nuevo, los dispositivos necesitarían una pantalla, pero además es necesario hardware dedicado para gráficos (un procesador de gráficos, también conocido como GPU) que, dependiendo del requerimiento necesario por el agente personificado, puede no ser lo suficientemente potente excepto dispositivos móviles de gama media-alta. A nivel de software, Android tiene soporte para la librerías de gráficos OpenGL [31] y en versiones más nuevas también para su evolución (la librería Vulkan) [32], las cuales permiten utilizar la GPU interna para generar gráficos tridimensionales en cualquier dispositivo Windows o basado en Unix, como Android [33] [34]. Para crear representaciones virtuales se podría desarrollar desde cero el motor que controlase al agente o utilizando motores gráficos actuales (como Unity o Unreal Engine, que incluyen soporte para Android) [35] [36], utilizando estándares desarrollados en proyectos externos para la evaluación y adaptación del agente, como puede ser el proyecto SAIBA, que busca estandarizar una estructura modular de procesamiento para representar, mediante agentes personificados virtuales, una interacción multimodal con el usuario, y que la generación de estos agentes virtuales sea rápida y flexible para servir las respuestas al usuario con eficiencia [37].

### 2.1.3. Servicios web para la generación de interfaces conversacionales (Dialogflow y otras alternativas)

En anteriores apartados, se ha definido la estructura de un sistema de diálogo hablado, explicando su estructura modular que, entre otras funciones, realiza un procesamiento del lenguaje natural utilizado para obtener un desglose semántico de la información recibida. Esta estructura puede ser desarrollada desde cero, pero hay varios servicios que ofrecen las funcionalidades necesarias para realizar este análisis de lenguaje natural interno necesario para entender el *input* del sistema y mantener un flujo de generación de respuestas nuevas que permiten que el sistema conversacional sea eficaz. Estos servicios ofrecen un análisis de intenciones de la información recibida, que descifra el significado de una frase y obtiene una colección de expresiones y

palabras clave que puedan ayudar a crear el contexto de la información recogida para, así, continuar el diálogo con respuestas coherentes y relevantes.

El primero de los servicios estudiados se llama **Dialogflow** (anteriormente llamado Speaktoit y Api.Ai) [6]. Dialogflow es un servicio web para la generación de agentes conversacionales, propiedad de Google desde 2016, que ofrece a sus usuarios la posibilidad de crear agentes conversacionales como los descritos en apartados anteriores, que reciban información de los usuarios, y mediante un modelo de lenguaje natural previamente definido por el desarrollador del sistema, sea capaz de responder al usuario con respuestas detalladas anteriormente en dicho modelo. Dialogflow ofrece soporte para el procesamiento de lenguaje natural de 18 idiomas, a los que se suma el reconocimiento independiente de 13 dialectos de dichos idiomas [38]. Aunque Dialogflow ofrece un plan gratuito (Standard Edition) para desarrolladores, también tiene un plan de pago por uso (Enterprise Edition) que ofrece más peticiones al servidor para interactuar con los agentes y más capacidad para utilizar servicios de Google relevantes para un agente conversacional como puede ser un servicio Speech-To-Text integrado. En la Tabla 2.2 se detallan los planes gratuitos y de pago de Dialogflow, mientras que en la Figura 2.9 se define el flujo de la información en Dialogflow:

| **Funciones** | **Standard Edition** | **Enterprise Edition** |
|---|---|---|
| Peticiones por texto | Peticiones ilimitadas gratis | Peticiones ilimitadas con un coste de 0.002 |
| Interacción por voz | Gratis hasta 1000 peticiones diarias, con un máximo de 15000 al mes | Peticiones ilimitadas (0.0065$ cada una) utilizando el servicio de Speech-To-Text de Google Cloud |
| Margen de tiempo de peticiones por texto | 3 peticiones/segundo | 10 peticiones/segundo |
| Acuerdo de nivel de servicio (Fiabilidad) | Ninguno | Más de un 99.9 % de tiempo activo al mes |
| Soporte | Por email y comunitario | Utilización del soporte de Google Cloud |
| Términos de servicio | Terminos de servicio de Dialogflow | Términos de servicio de Google Cloud |

Tabla 2.2: Comparación de planes de Dialogflow [6]

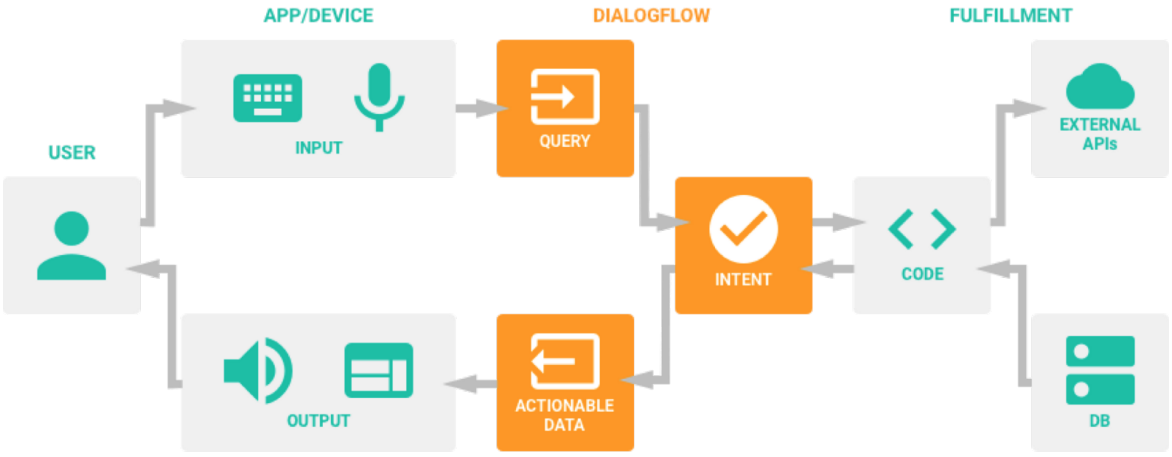


Figura 2.9: Flujo de información en Dialogflow [39]

En la anterior Figura 2.9, se refleja una explicación básica del flujo de información que se encontraría al interactuar con Dialogflow. En una interacción básica con un agente definido en el servicio, existe el rol del usuario que, mediante texto o reconocimiento de voz, contacta con la API del agente (en la figura, lo que llama *query* o, en castellano, consulta). La información enviada por el usuario entra en Dialogflow, y se procede a realizar un análisis de dicha información, con el fin de reconocer el significado y las palabras claves del mensaje transmitido por el usuario. Una vez encontrados estos parámetros, el agente de Dialogflow coteja esta información con el modelo de procesamiento de lenguaje natural creado por el desarrollador, con el objetivo de encontrar la intención del usuario (lo que en la figura se llama *intent*). Este *intent* definido previamente es capaz de realizar lógica también previamente establecida por el desarrollador, y, con la intención de determinar una respuesta con la información más adecuada para entregársela al usuario, es capaz de consultar servicios externos de información (bases de datos, APIs externas) con las que complementar las respuestas previamente definidas en el modelo del agente. Este proceso se llama *fulfillment*, o rellenado, y cada *intent* tiene un plan de rellenado distinto para adecuarse a las consultas de los usuarios. Una vez conseguida información relevante para las peticiones del usuario, el intent utiliza esta información para rellenar respuestas definidas en el modelo del agente, que son enviadas al usuario como respuesta a la petición realizada a la API del agente. Esta respuesta, posteriormente, puede ser mostrada al usuario tanto por texto como por síntesis de voz, dependiendo de las capacidades del dispositivo en el que se esté ejecutando la interacción con el cliente.

La definición de intenciones, o *intents*, dentro del agente de Dialogflow se realiza con la utilización de frases de entrenamiento (*Training phrases*). Estas frases son textos base, escritos con lenguaje natural, que representan la entrada esperada de un usuario para ese *intent*, expresadas de distintas maneras para así tener variedad de ejemplos para que el reconocimiento de intenciones sea más preciso. Dialogflow, mediante un sistema integrado de aprendizaje que utiliza tanto algoritmos de aprendizaje automático desarrollados por Google como aprendizaje manual realizado por el desarrollador

del agente [40], es capaz de adaptar el *intent* para reconocer frases que, aunque no hayan sido definidas palabra por palabra en estas frases de entrenamiento, puedan ser similares en significado y contenidos a las frases de entrenamiento definidas por el desarrollador, y así añadir flexibilidad y precisión al agente a la hora de reconocer las intenciones del usuario. En la definición de estas frases de entrenamiento también se determina la existencia de palabras clave que puedan ser encontradas en una entrada (por ejemplo, en un agente al que los usuarios preguntan datos meteorológicos, en las frases de entrenamiento se puede determinar que es de esperar que el usuario utilice palabras clave como el momento o la localización de la pregunta que está realizando). Estos conceptos clave se llaman internamente entidades, o *entities*. Dialogflow viene con un diccionario integrado de términos comúnmente utilizados con significado genérico, como números, ciudades, fechas, géneros artísticos y más temas de uso general que, para una variedad de casos, puedan ser potencialmente útiles para el agente, pero Dialogflow también incluye la posibilidad de definir nuevos tipos de entidades, definidos por un diccionario de palabras escritas por el desarrollador, y que posteriormente pueden ser utilizados dentro de cualquier frase de entrenamiento para concretar nuevos términos a buscar dentro de la entrada de un *intents*. Todos los tipos de entidad son definidos como variables dentro del agente, y estas pueden ser utilizadas posteriormente no solo para el análisis y generación de respuestas en su *intent* correspondiente, si no como banco de memoria de los términos utilizados por el usuario durante la conversación con el agente, y así poder utilizar dichos términos para complementar las respuestas de otros *intent*, lo que dentro de Dialogflow se llama contexto de intención (o *intent context*) [41] [42]. En la Figura 2.10 se muestra la interfaz de Dialogflow para definir las frases de entrenamiento de un *intent*:

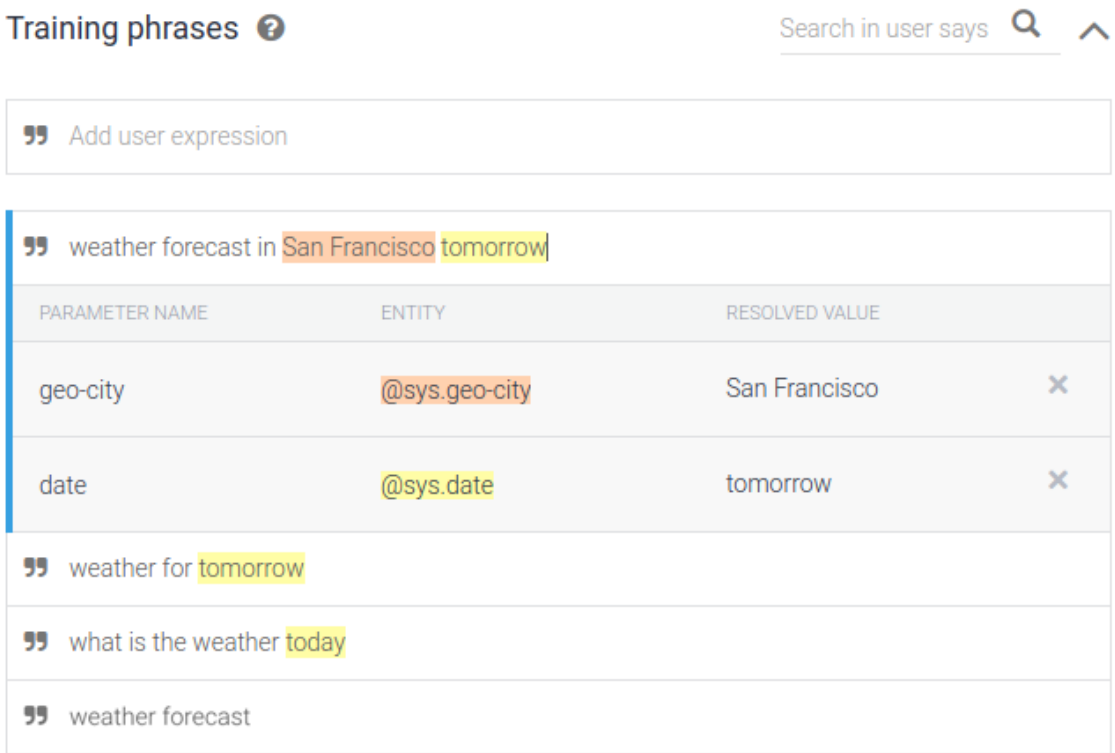


Figura 2.10: Ejemplo de desarrollo de *training phrases* de un *intent* en Dialogflow [43]

Los *intents*, una vez reconocidos, son capaces de acceder a información externa al agente para complementar la información necesaria para generar una respuesta detallada al usuario. Este proceso, tal y como fue explicado anteriormente, se llama rellenado o *fulfillment*, y se trata de una petición HTTP que el agente realiza a un servidor externo, potencialmente creado por el desarrollador del agente, y que se encarga de añadir lógica al funcionamiento del agente ante la aparición de este determinado *intent*. Esta lógica puede dar funcionalidades extra al agente que Dialogflow no es capaz de realizar directamente, como pueden ser peticiones a bases de datos externas con información relevante para la intención del usuario, y que pudiera ser importante para la definición de respuestas. El servidor, una vez realizado el procesamiento definido en su programación, le devuelve, con otra petición HTTP, la información encontrada al agente de Dialogflow. Este paso, para el funcionamiento del agente, es opcional, pero ofrece posibilidades superiores a las encontradas en el funcionamiento estándar del agente de Dialogflow [44].

Una vez encontrada la intención del *input* del usuario, habiendo definido los términos clave utilizados por este, y potencialmente, haber realizado consultas externas para complementar la información obtenida, el agente es capaz de generar una respuesta al usuario. Estas respuestas utilizan un texto base definido por el desarrollador del agente, pero que complementan con el estudio realizado en los pasos anteriores. El agente, en el momento de generar la respuesta, tiene disponibles todas las entidades encontradas en el *input* de la conversación, y estas pueden ser reutilizadas como variables en la respuesta que el agente da al usuario. Además, toda la información recibida en el proceso de *fulfillment* también puede ser utilizada para rellenar las respuestas, de nuevo como variables que pueden complementar la formación de la respuesta. En la Figura 2.11, se puede ver un ejemplo de la definición de respuestas por parte del desarrollador, y como estas pueden utilizar las variables de las entidades y el *fulfillment* para concretar la información enviada de vuelta al usuario.

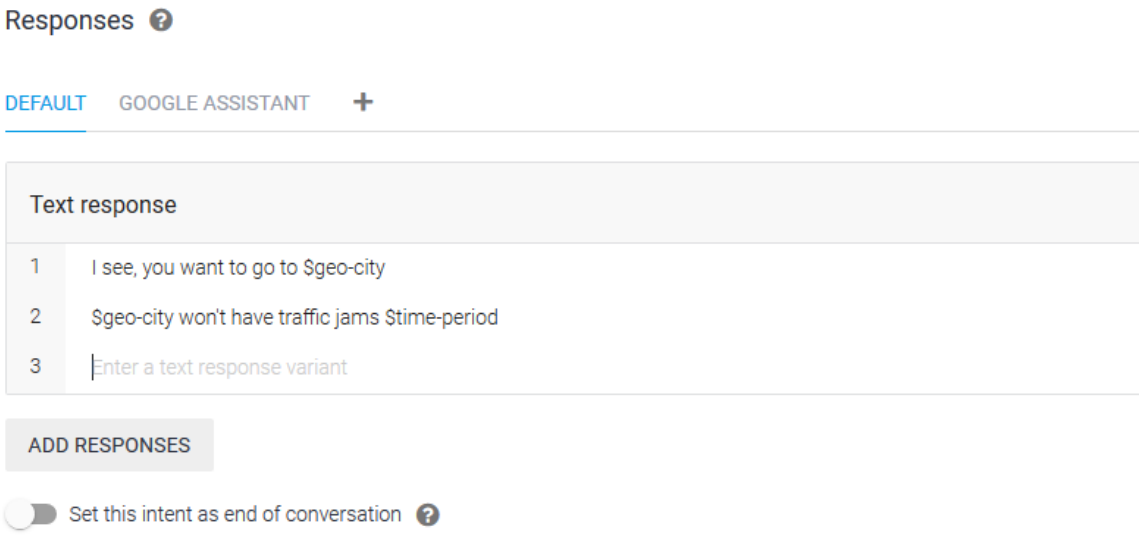


Figura 2.11: Ejemplo de respuesta de un *intent* en Dialogflow

Por último, concretar que Dialogflow puede ser integrado en cualquier plataforma que permita el envío y recibimiento de información mediante protocolos HTTP, pero hay varios SDKs (*Software Development Kits*) disponibles para su libre uso en varias plataformas, como, por ejemplo, Android [45]. La integración en Android utilizando el SDK de código abierto que ofrece Dialogflow ofrece la posibilidad de utilizar métodos y funciones previamente definidas en este, que contienen la funcionalidad básica necesaria para realizar peticiones a un agente concreto y recibir respuestas de este. Las respuestas se reciben en un formato JSON estándar, pero para enviar información, se pueden utilizar dos formatos:

- **Texto**. Haciendo una petición a Dialogflow con un cuerpo de texto plano la introducción de información se realizaría directamente en el formato con el que el agente de Dialogflow trabaja y realiza sus análisis. Este texto puede generarse en Android o bien escribiendolo con teclado o utilizando el reconocimiento de habla explicado en apartados anteriores.
- **Reconocimiento de habla de Dialogflow**. El SDK de Dialogflow para Android contiene un sistema de grabación de audio integrado, independiente del de Android. Con esta funcionalidad, el SDK busca evitar un paso previo al desarrollador de la aplicación para introducir la información del usuario, puesto que con este archivo de audio los servidores de Dialogflow realizan un proceso de Speech-To-Text que acaba ofreciendo al agente el texto resultante de esta conversión.

Aparte de la utilización de los SDKs, Dialogflow también ofrece la posibilidad de activar integraciones prediseñadas para utilizar el agente dentro de aplicaciones y asistentes virtuales ya existentes que hacen de interfaz con Dialogflow, como si fuese un agente interno dentro de estas apps, sin necesidad de desarrollar una aplicación de integración aparte. Algunas de las integraciones disponibles son con Google Assistant, Slack, Telegram o Twitter, todas ellas aplicaciones utilizadas diariamente por una amplia multitud de usuarios, y que permiten a los desarrolladores de agentes tener un amplio marco de potenciales usuarios para desplegar sus agentes y todas sus funcionalidades.

Por otro lado, otro servicio web para generar este tipo de interfaces conversacionales fue creado por otra empresa multinacional, IBM, en 2016, llamado **Watson Assistant** [46]. Watson Assistant es un servicio de procesamiento de lenguaje natural que utiliza las capacidades de aprendizaje automático de la plataforma Watson de IBM, un sistema computacional basado en inteligencia artifical desarrollado desde 2004 por parte del equipo de investigación de IBM [47]. Varias capacidades de Watson, como sus servicios de identificación de imágenes o varios servicios de análisis de lenguaje natural, son accesibles a los consumidores gracias a la plataforma IBM Cloud, su plataforma de distribución de servicios en la nube, y Watson Assistant es uno de ellos. Watson Assistant, de la misma manera que Dialogflow, ofrece a los desarrolladores la capacidad de crear agentes conversacionales que interactúen con una entrada de información procedente del usuario, recojan toda la información disponible de esta entrada y de elementos externos al agente, y devuelvan al usuario una respuesta coherente respecto a la entrada recibida. De momento, el servicio está disponible para 13 idiomas, pero el

soporte para algunos de ellos está en beta y su utilización no es precisa [48]. Watson Assistant ofrece varios planes disponibles para los usuarios: uno gratuito (*Lite*) con capacidades limitadas, otro para desarrolladores que esperen más capacidad de utilización del servicio (llamado Estándar) y finalmente un plan dedicado a desarrolladores y organizaciones con necesidad de ofrecer una funcionalidad personalizada a sus usuarios y seguridad extra dentro de las instancias del servicio, llamado *Premium*. En la Tabla 2.3, se comparan los planes gratuitos y estándar, ya que el *Premium* es personalizable:

| **Funciones** | **Lite** | **Estándar** |
|---|---|---|
| Peticiones | 10000 peticiones/mes | Peticiones ilimitadas por 0.00188€/petición |
| Espacios de trabajo máximos | 5 espacios | 20 espacios |
| Intenciones (*intents*) máximas | 100 *intents* | 2000 *intents* |
| Entidades (*entities*) máximas | 25 entidades | 1000 entidades |

Tabla 2.3: Comparación de planes de Watson Assistant [49]

En la Figura 2.12, se puede ver el flujo de la información básico de un diálogo con un agente de Watson Assistant, y se puede observar como el flujo es considerablemente similar al equivalente de Dialogflow (Figura 2.9):

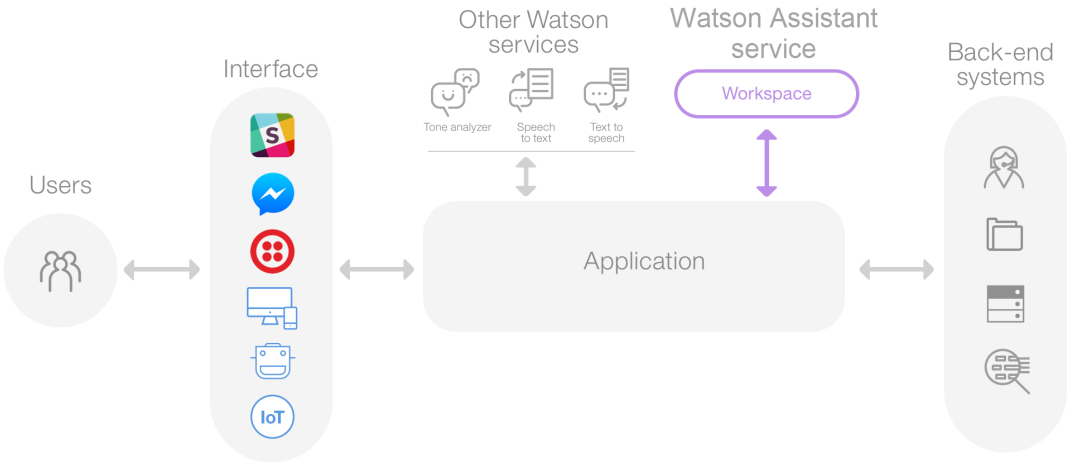


Figura 2.12: Flujo de información en Watson Assistant [50]

Este flujo realiza operaciones similares al flujo equivalente de Dialogflow, y el orden de las operaciones es también igual. Para interactuar con Watson Assistant, los usuarios deben utilizar la aplicación de integración creada por el desarrollador, que contacta con el agente correspondiente de Watson Assistant y le envía la información definida por el usuario como entrada en un formato de texto o de archivo de audio

(que luego tendrá que ser analizado por servicios de Speech-To-Text de IBM Cloud). Posteriormente, Watson Assistant realizará un análisis de la entrada, que, junto a información recogida de servicios externos (y otros servicios de IBM Cloud, concretará el contexto en el que se encuentra la entrada para, así, generar una respuesta que irá de vuelta al usuario por el mismo canal que la entrada. Como se puede ver, la gran diferencia con el flujo de Dialogflow es la capacidad de utilizar servicios adicionales de IBM Cloud dentro del funcionamiento del agente, y así ofrecer una experiencia más adecuada y personalizada a los usuarios del sistema.

Sin embargo, internamente, el funcionamiento de Watson Assistant difiere del de Dialogflow. Aunque ambos utilizan términos y conceptos similares para crear el modelo del agente, Watson Assistant ofrece algunas variaciones en la definición de las intenciones y las potenciales conversaciones. El procedimiento inicial para definir el modelo del agente comienza con la definición de intenciones (*intents*) que quieran ser utilizadas para reconocer la respuesta del usuario. En un *intent*, el desarrollador del agente define las frases de entrada esperadas por parte del usuario, al igual que marcando las palabras clave (o entidades, *entities*) que espera recibir en ellas. Al igual que en Dialogflow, las entidades se consideran variables que pueden ser utilizadas en otras partes del agente, y además de una entidades definidas previamente por IBM, el usuario puede definir sus propios tipos de entidades manualmente, introduciendo conceptos y sinónimos que quiera que sean reconocidos dentro de ese tipo de entidad. Además de poder definirlas manualmente, el desarrollador tiene la opción de importar paquetes de intenciones y entidades previamente definidas en IBM, que contienen modelos básicos acerca de temas comunes por el usuario medio, como pueden ser intenciones y términos bancarios o de servicio de atención al cliente. En la Figura 2.13 se muestra la interfaz de creación de *intents* en Watson Assistant:

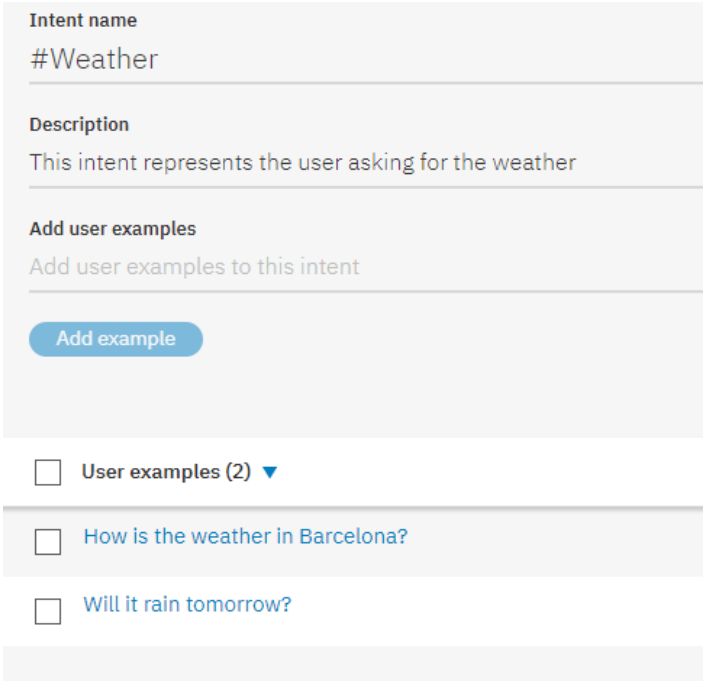


Figura 2.13: Ejemplo de creación de un *intent* en Watson Assistant

Una vez identificados los *intent* y las entidades, el agente completa la información obtenida con datos de servicios externos, tanto de IBM Cloud como no integrados en este. Realizando estas peticiones, el agente complementa la información con información no contemplada dentro del agente, como pueden ser datos de APIs externas o de bases de datos, al igual que Dialogflow. Una vez conseguida esta información complementaria, el agente debe generar una respuesta. La gran diferencia con Dialogflow radica en este sistema, puesto que mientras que en Dialogflow se definen respuestas por cada intención creada, Watson Assistant contiene un sistema de definición de conversaciones, donde el desarrollador del agente diseña, mediante la colocación de nodos y flechas, el flujo de diálogo esperado por el agente, y las respuestas a generar cuando se encuentra con determinadas intenciones o identidades. Dialogflow tiene un sistema similar para recuperar información de otros *intent* para crear una conversación fluida (usando contextos como fue explicado anteriormente), pero el sistema de Watson Assistant ofrece una representación visual de este proceso que, potencialmente, puede ser más eficaz para los desarrolladores de agentes. En la Figura 2.14 se puede ver como se definen estos diálogos y sus nodos en la interfaz de Watson Assistant:

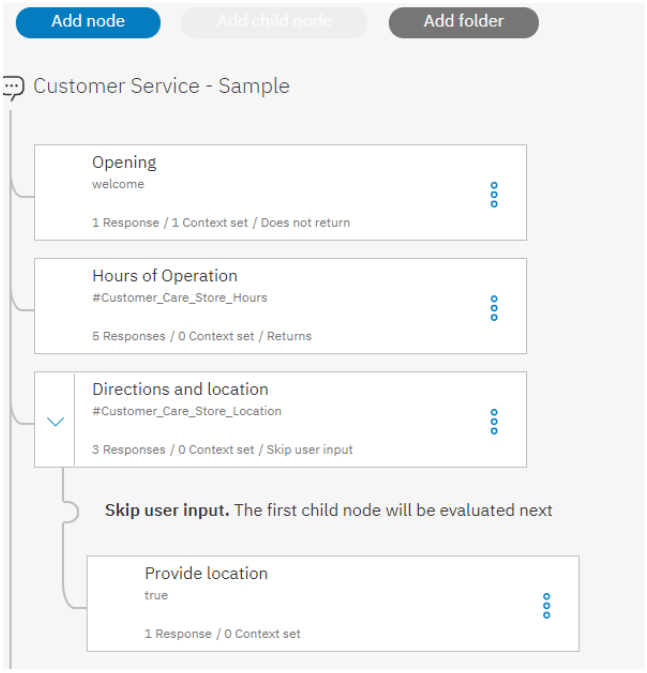


Figura 2.14: Ejemplo de creación de un diálogo en Watson Assistant

Por último, destacar que Watson Assistant ofrece, al igual que Dialogflow, la posibilidad de utilizar SDKs para integrar la interacción con los agentes en aplicaciones externas que puedan ser utilizadas por los usuarios en sus dispositivos. Estos SDKs están disponibles en múltiples plataformas, como Android, y son de código abierto [51]. El SDK de Android permite la utilización de sistemas de Speech-To-Text integrados en IBM Cloud, activando el micrófono del dispositivo para grabar un archivo de audio que, posteriormente, es enviado a los servidores de IBM y procesado internamente, y también la posibilidad de utilizar texto plano para realizar la consulta mediante una petición HTTP. Además, se ofrece la posibilidad de realizar una integración directa en Slack y Facebook Messenger.

## 2.2. Análisis de sentimientos

El análisis de sentimientos, también llamado *sentiment analysis*, es un subproceso de la minería de textos (proceso basado en la recuperación de información de un texto mediante el uso de tecnologías como aprendizaje automático o estadística), encargada de estudiar un fragmento de texto y reconocer el tono subjetivo con el que se ha formado. Este tono, regularmente definido como un valor positivo, negativo o neutro llamado polaridad, representa un concepto relevante en la identificación del contexto y significado de un texto escrito en lenguaje natural, dando la información necesaria para entender la actitud del creador de dicho texto a la hora de transmitir la información representada en el fragmento [52]. El análisis de sentimientos puede ser utilizado en muchos entornos, pero, actualmente, es utilizado principalmente para conseguir una imagen definida de la intención subjetiva del creador con el texto formulado, y esta imagen puede ser utilizada como medida de valoración de la opinión de este usuario sobre temas concretos. Es por ello que es usado por un alto número de empresas, que recogen un número de opiniones sobre productos suyos por parte de los consumidores (desde varias fuentes, como críticas, redes sociales o incluso llamadas telefónicas), y las estudian con sistemas de análisis de sentimientos automáticamente para generar un estudio de la valoración del producto por parte del público [53]. Este análisis de mercado es una de las aplicaciones con mejores resultados en las que se integra una forma de *sentiment analysis*. Según estudios realizados, la capacidad de poder hacer un análisis de sentimientos de las críticas que los consumidores generan sobre ciertos productos, ayudan a descubrir cuales son los elementos que más les importan a los usuarios sobre el producto, y así discernir con un análisis automático la impresión de los consumidores no solo del producto en si, si no de cada una de sus cualidades, y así tener un feedback creado automáticamente certero [54]. Con un feedback así, también se han generado proyectos que utilizan este análisis de sentimientos para crear sistemas de recomendación adaptados a los usuarios, que, conociendo los elementos importantes para este según sus críticas a otros productos, filtran una lista de productos recomendados que potencien las cualidades requeridas y deseadas por el cliente, consiguiendo un aumento de la satisfacción de este [55].

El procedimiento para hacer un análisis de sentimientos implica la utilización de técnicas de modelado estadístico, que pueden ser desde reglas simples definidas anteriormente con patrones a buscar en los textos de entrada hasta diccionarios de términos claves, que ponderan la polaridad hasta conseguir un resultado definitivo que represente la intención emocional del texto. Actualmente, muchos algoritmos de *sentiment analysis* también hacen uso de técnicas de aprendizaje automático para entrenar estos modelos, y que puedan adaptarse a distintas entradas de texto, modificando sus patrones y diccionarios de términos automáticamente para aumentar la precisión de la generación de la polaridad en el sistema. Algunos ejemplos de algoritmos con capacidades de aprendizaje automático similares se pueden encontrar en varios proyectos de investigación, como un sistema de análisis de sentimientos que es capaz de estudiar contenido de Twitter [56], o el estudio de los chats integrados en videojuegos con capacidades multijugador por Internet, en busca de indicios de acoso y *cyberbulling* por parte de los jugadores [57]. Un sistema de análisis de sentimientos puede recibir textos de cualquier longitud, regularmente el sistema hace el análisis separando el tex-

to original en fragmentos con lógica sintáctica y estudiándolos primero por separado, para luego analizarlos juntos, pero los algoritmos de análisis de sentimientos, al tener que trabajar a un nivel también contextual, funcionan de una manera más precisa con textos cortos, que contengan únicamente la información básica que se quiere estudiar, en lugar de textos largos que, aunque contienen la misma información, contienen ruido que dificulta el análisis, y también hacen que un modelo basado en aprendizaje automático se vea perjudicado y desajustado [58].

En la Figura 2.15, se puede observar el orden de los procesos que sigue un sistema de análisis de sentimientos estándar:


Figura 2.15: Flujo de procesos en el análisis de sentimientos [59]

Como se puede observar en la anterior figura, hay 7 pasos básicos que un sistema de *sentiment analysis* sigue para acabar consiguiendo una representación del sentimiento del texto, definido en la polaridad del texto. Los pasos, explicados con más detalle, son los siguientes:

- **Introducción de texto (*Text Input*)**. En este paso, un usuario u otro sistema introduciría el texto que quiere analizar. Este texto, dependiendo del sistema, podría ser introducido de varias maneras, como por ejemplo una petición HTTP con el contenido deseado o mediante la introducción de sistemas de reconocimiento del habla.
- **División (*Tokenization*)**. Aquí el sistema dividirá el texto original en subdivisiones (*tokens*) con lógica sintáctica (frases o expresiones), que a su vez se dividirá en cada una de las palabras que contengan estas subdivisiones, para que el sistema pueda trabajar con cada una de las partes por separado, y no depender, de momento, de un contexto real.

- **Filtrado de palabras vacías (*Stop Word Filtering*)**. Los idiomas humanos, por la necesidad de crear conexiones entre los elementos clave de una frase (como los sustantivos y los verbos) y así construir estructuras fluidas y consistentes, hacen uso de formas gramaticales sin significado alguno, pero que sirven para cumplir la función anterior. En el castellano, por ejemplo, se hace uso de elementos como los artículos, los pronombres o las preposiciones, que no añaden información relevante al mensaje del texto. Dentro de la minería de texto, estas palabras se llaman palabras vacías (*stop words*), que, de cara al sistema de análisis, solo introducen ruido en el texto a estudiar, y son evitables. Utilizando filtros de palabras vacías, un sistema de análisis de sentimientos eliminaría estas palabras vacias de su lista de *tokens*, y evitaría que el análisis se viese perjudicado.

- **Manejo de la negación (*Negation Handling*)**. La negación, dentro de cualquier lenguaje, implica un cambio del significado de todo lo que lo rodea. En un idioma humano, la negación convierte el significado de una frase en el completo contrario, y lo mismo pasa en lenguajes basados en la lógica, como puede ser un lenguaje de programación, donde la negación implica que la lógica se convierte en la contraria. Es por ello que, en el análisis de sentimientos, que se realiza sin mirar en un primer momento más allá de las palabras, la negación podría cambiar radicalmente la polaridad de un texto. En el análisis de sentimientos es necesario reflejar esto, y en este paso de manejo de la negación, el algoritmo recoge las entidades de negación encontradas en el texto y las une a los *tokens* correspondientes para darles contexto, y así evitar que el sistema malinterprete el significado del texto original.

- **Búsqueda de procedencia (*Stemming*)**. En esta fase, el algoritmo descompone cada una de las palabras encontradas hasta llegar a su raíz (llamada en inglés *stem*). Teniendo las raices, el algoritmo es capaz de buscar relaciones entre palabras del texto, y así ir construyendo el contexto del texto original por la relación de las palabras entre ellas.

- **Clasificación (*Classification*)**. Con todos los *stems* y las relaciones entre ellos, el sistema comienza a ir analizando la polaridad de cada uno de los elementos disponibles, reconstruyéndolos y analizándolos en el orden inverso en el que los dividió (es decir, yendo primero por las palabras, luego las frases).

- **Clase del sentimiento (*Sentiment Class*), o salida**. Esta última fase recoge las clasificaciones hechas en el paso anterior para cada una de las estructuras reformadas, y pondera la polaridad de todas ellas para conseguir una polaridad final del texto completo. Con esta polaridad, el sistema devuelve al usuario (u otro sistema) una respuesta que contiene esa información, terminando así el análisis a este texto y pasando a un estado de espera de otro texto (volviendo al primer paso, la introducción de texto).

### 2.2.1. Servicios de análisis de sentimientos

Al igual que con los sistemas de diálogo hablado, el desarrollo de un sistema de análisis de sentimientos puede ser desarrollado desde cero, pero el entrenamiento del

modelo puede llegar a ser muy complicado y costoso. Es por ello que, actualmente, hay una variedad de servicios online que ofrecen la posibilidad de dar un texto plano como entrada y, como respuesta, devuelven el análisis realizado al texto de entrada. Estos servicios son útiles para desarrolladores que quieren utilizar un análisis de sentimientos dentro de sus sistemas, ya que funcionan mediante el uso de APIs que utilizan peticiones HTTP para mandar información del sistema al servicio de análisis y viceversa. A continuación, en este apartado, se detallarán tres servicios regularmente utilizados para hacer análisis de sentimientos, pero es solo una selección de una gran cantidad de servicios disponibles online [60].

El primer servicio estudiado se llama **MeaningCloud** [61]. Nacido en 2015, MeaningCloud ofrece a sus usuarios la posibilidad de utilizar servicios de minería de texto, entre los que se encuentra el análisis de sentimientos de los textos. Además del *sentiment analysis*, MeaningCloud tiene la capacidad de clasificar, subdividir en temas e identificar los elementos clave de los textos, gracias a su motor de minería de texto. Su servicio de análisis de sentimientos es capaz de reconocer 6 idiomas, y de ofrecer tanto la polaridad general del texto como de cada uno de sus componentes, además de un estudio de los conceptos clave del texto, devolviendo todo al usuario que hace la petición. MeaningCloud ofrece una cantidad de planes para desarrolladores muy amplia, que van desde una cuenta gratuita hasta un plan para empresas personalizable tanto en precio como en condiciones. En la Tabla 2.4, se detallan cada uno de los planes del servicio.

| **Funciones** | **Free** | **Start-Up** | **Professional** | **Business** | **Enterprise** |
|---|---|---|---|---|---|
| Peticiones/mes | 20000 | 120000 | 700000 | 4200000 | Personalizable |
| Peticiones/seg. | 2 | 5 | 10 | 15 | Personalizable |
| Soporte | Sin soporte | Estándar | Estándar | Estándar | Premium |
| Precio mensual | Gratis | 99$ | 399$ | 999$ | Personalizable |

Tabla 2.4: Comparación de planes de MeaningCloud [62]

MeaningCloud, como la gran mayoría de estos servicios, realizan la interacción con sistemas externos mediante peticiones HTTP. El usuario (o su aplicación) envía una petición HTTP a la API de MeaningCloud conteniendo el texto a analizar y una lista de opciones configurables que ajustarán el análisis una vez recibido en el servicio, que realiza el análisis internamente, y devuelve a la aplicación un mensaje en formato JSON o XML que contiene información global del texto, como su polaridad, su subjetividad o el ratio de confianza que tiene en el análisis realizado, pero también contiene un desglose de todas las divisiones realizadas en el texto, que contienen su propio análisis también tratando los campos descritos anteriormente. La polaridad en este análisis se devuelve al usuario como 6 posibles valores: P+, P, NEU, N y N+ (subjetivo) o NONE (objetivo), que representan una polaridad, según ese orden, de muy positiva a muy negativa. Además del análisis estándar, MeaningCloud ofrece la posibilidad de crear diccionarios de términos y expresiones personalizadas que puedan ser utilizadas dentro del análisis realizado por el sistema, pudiendo definir la polaridad de dichas expresiones

por separado y así poder realizar un análisis más concreto para temas determinados por los desarrolladores. Respecto a la integración del servicio con otros sistemas, cualquier plataforma con acceso a Internet puede utilizarlo, pero también MeaningCloud ofrece tres SDKs (para Python, PHP y Java) que ayudan a que la integración con sistemas de utilicen estos lenguajes sea más sencilla para el desarrollador. Además, también ofrece extensiones para Microsoft Excel y RapidMiner, entre otros [63].

Por otro lado, otro servicio también seleccionado de análisis sentimientos es otro producto de IBM, parte de la plataforma Watson encontrada en su catálogo de servicios en la nube IBM Cloud. Este servicio se llama **Watson Natural Language Understanding (o Watson NLU)** [64], y es un servicio que es capaz de realizar varios procedimientos de minería de texto sobre cualquier entrada de texto plano que llegue a su API, utilizando la inteligencia artificial contenida en Watson. En la Tabla 2.5, se comparan los planes disponibles en el servicio, que van desde un plan gratuito pero limitado a planes *premium* con soporte dedicado y personalizable por los desarrolladores.

| **Funciones** | **Lite** | **Estándar** | **Premium** |
|---|---|---|---|
| Peticiones/mes | 30000 | 0.002257€hasta 250000 peticiones, 0.000752€hasta 5000000 peticiones y 0.00015€desde 5000001 peticiones | Personalizable |
| Modelos personalizados | 1 | 602€/ modelo | Personalizable |
| Soporte | Soporte básico | Estándar | Premium |

Tabla 2.5: Comparación de planes de Watson NLU [65]

Watson NLU es un servicio de minería de datos que, al igual que MeaningCloud, tiene varios subservicios con los que los desarrolladores pueden interactuar. Aparte del análisis de sentimientos anteriormente definido, Watson NLU permite hacer un análisis de categorización del texto, utilizando uno de los múltiples modelos con los que viene integrado (por ejemplo, novelas, marcas de coches, términos médicos, etc.) para determinar qué es cada elemento del texto encontrado, además de una descripción del contexto del texto completo (por ejemplo, inteligencia artificial, deportes, etc.). Por otro lado, el análisis del sentimiento se puede realizar de dos maneras. La primera es un análisis emotivo, un tipo de análisis de sentimientos más avanzado que no devuelve una polaridad, si no que determina, dentro de una de 5 emociones básicas (felicidad, tristeza, miedo, asco y enfado), el tono del texto introducido. La segunda manera es el análisis de sentimientos explicado anteriormente, que devuelve la polaridad del texto completo, o de palabras concretas que el desarrollador quiera marcar como palabras objetivo. En este caso, la polaridad viene definida del -1 (extremo negativo) al 1 (extremo positivo). Watson NLU ofrece la posibilidad a los desarrolladores de utilizar tres SDKs previamente desarrollados por IBM, disponibles para NodeJS, Python y Java, además de poder los SDKs genéricos de todos los servicios de Watson en IBM Cloud o utilizar peticiones HTTP regulares para contactar con el servicio [66].

Por último, el último servicio estudiado en este apartado es el llamado **Aylien** [67]. Aylien, al igual que los dos servicios anteriores, es un servicio de minería de texto que ofrece un alto número de posibilidades de análisis basados en entradas de texto disponibles para los desarrolladores. Entre las opciones ofrecidas a los desarrolladores, Aylien ofrece cinco planes para ajustar los recursos utilizados por una aplicación concreta dentro del servicio. Estos planes se detallan en la Tabla 2.6:

| **Funciones** | **Free** | **Micro** | **Small** | **Medium** | **Large** |
|---|---|---|---|---|---|
| Peticiones/mes | 30000 | 80000 | 180000 | 2400000 | 5400000 |
| Soporte | Ninguno | Email | Email | Email | Email y teléfono |
| Precio mensual | Gratis | 49$ | 199$ | 649$ | 1399$ |

Tabla 2.6: Comparación de planes de Aylien [68]

El funcionamiento de Aylien es muy similar al resto de servicios de análisis de sentimientos. Aylien ofrece la capacidad de, utilizando su API, realizar análisis de varios aspectos de la minería de textos (extracción de conceptos, clasificación del texto, etc.). En este caso, el servicio de análisis de sentimientos ofrece cualidades similares a la de los otros dos, al introducir un texto a través de su servicio, este te devuelve una polaridad (representada con un indicador positivo o negativo) junto con un ratio de confianza relacionado con dicha polaridad, que va del 0 (poca confianza de haber realizado el análisis bien) a 1 (análisis muy preciso). Además, el mismo análisis te da un indicador de subjetividad, para comprender si el texto introducido es objetivo o subjetivo en su construcción. Aylien ofrece un segundo tipo de *sentiment analysis*, que, en vez de tratar con el texto completo, divide el texto en elementos separados y hace un análisis de sentimientos de cada uno de ellos, además de buscar un dominio para estos elementos. Actualmente, en este último tipo de análisis, Aylien solo permite el reconocimiento de textos acerca de hoteles, restaurantes, coches y aerolíneas, por lo que no es una opción precisa para cualquier mensaje que no entre dentro de uno de esos cuatro campos. Además, al contrario que los otros dos, Aylien no permite crear modelos personalizados para ajustar el análisis un tema definido por el usuario, por lo que se pierde algo de precisión respecto a los otros dos. Por último, destacar que Aylien ofrece la facilidad a los desarrolladores de utilizar cualquiera de sus siete SDKs disponibles para realizar una conexión con el cliente más sencilla de programar.

A pesar de solo haber estudiado a fondo tres servicios, se puede ver que, por lo general, todos los servicios de análisis de sentimientos ofrecen unas funcionalidades muy similares entre ellos, y que la diferencia entre ellos se basa en la precisión de los modelos utilizados internamente por cada uno de ellos. Todos ellos utilizan actualmente modelos basados en aprendizaje automático, por lo que el ajuste del modelo es más preciso que en un modelo puramente estadístico, y podrán dar buenos resultados ante cualquier texto [69].

## 2.3. Procesos de selección laborales

Un proceso de selección laboral es un procedimiento, llevado a cabo por una o varias empresas, con el objetivo de encontrar una serie de candidatos a un puesto de trabajo, realizar un estudio de varias de sus competencias en relación a dicho puesto, y proceder, mediante varias fases, a filtrar candidatos hasta encontrar al candidato que más se adecue al puesto ofrecido. Este proceso se compone de varias fases necesarias para que el departamento de Recursos Humanos (RRHH) pueda realizar los filtros necesarios y las comprobaciones necesarias para obtener una imagen de los candidatos que puedan ayudar a elegir al candidato perfecto. Las fases son las siguientes [70]:

- **Planificación**. En esta fase, Recursos Humanos realiza un estudio de las necesidades del puesto ofrecido, que capacidades deben ser destacadas en los candidatos, y cuantas pruebas se les realizarán.
- **Reclutamiento y preselección de candidatos**. Aquí se oferta la vacante mediante varios canales, que puedan atraer a candidatos interasados en dicha vacante para que envien sus curriculms vitae. Una vez recibidas todas las peticiones, el departamento de Recursos Humanos hace una preselección de los candidatos que puedan ser interesantes de conocer en el proceso de selección.
- **Realización de pruebas**. En esta fase, los candidatos se someten a una variedad de pruebas diseñadas para destacar sus competencias en los elementos clave necesarios para el puesto. Estas pruebas pueden ser de una multitud de tipos, desde exámenes escritos (de lógica, matemáticas, orientación espacial, comprensión lectora...) a dinámicas grupales donde varios candidatos se juntan para realizar una prueba en grupo, destacando habilidades de trabajo en equipo y personales. Esta fase es opcional, y, aunque la siguiente fase es también una prueba, es una constante en todos los procesos de selección actuales.
- **Entrevista de trabajo**. La entrevista de trabajo es la prueba esencial en todos los procesos de selección. Realizada normalmente en un carácter presencial, la entrevista de trabajo sitúa a un solo candidato en un cuestionario, realizado por uno o más entrevistadores, donde es preguntado por múltiples aspectos de su carrera laboral y personal, como pueden ser estudios anteriores, aptitudes o cuestiones de carácter práctico. Con estas preguntas, los entrevistadores buscan encontrar las cualidades previamente definidas como importantes en dicho candidato, preguntándole directamente. Estas entrevistas pueden realizarse varias veces, por varios entrevistadores que puedan concretar ciertos detalles respecto a entrevistas anteriores. En muchas ocasiones, este es el último paso realizado por los candidatos en el proceso de selección.
- **Valoraciones y decisión**. La última fase, para los departamentos de RRHH, es elaborar un estudio de todos los candidatos que han llegado a la última fase del proceso, para que, finalmente, el encargado de realizar la contratación decida a cuál de ellos contratar.

Como se puede ver, en este proceso hay una fase que destaca por encima del resto: la entrevista de trabajo. La entrevista, dependiendo del proceso de selección, puede ser

el único contacto de los candidatos con la empresa, y, por ello, la única oportunidad de conseguir el puesto disponible. Es por ello que el proceso de entrevistas suele ser una situación estresante para los candidatos, y uno que no se puede ensayar. Las entrevistas, dependiendo de la planificación seguida, pueden ser muy diferentes incluso dentro de una misma vacante, ya que se ajustan al perfil del candidato a entrevistar aparte de al propio puesto. Sin embargo, por lo general las entrevistas tienen una estructura definida, muchas veces basada en el orden del curriculum del candidato, pero con algunas variaciones [71]:

- **Saludos e introducción**. El entrevistador y el candidato se presentan, y el entrevistador explica el formato de la entrevista al otro. En esta sección también hay preguntas menos relacionadas con el proceso de selección, que ayudan a que el candidato se encuentre en un estado tranquilo, lo que coloquialmente se llama romper el hielo.
- **Preguntas del entrevistador**. En este momento, el entrevistador comienza a plantear preguntas al candidato, que debe responderlas con sinceridad. Algunas de estas preguntas se generan al surgir dudas en datos del curriculum, pero muchas preguntas se plantean para conocer el pensamiento del candidato ante ciertas situaciones. Normalmente, esta sección no tiene una estructura definida, pero hay algunas guías de referencia para entrevistadores para que puedan crear una estructura definida [72]:
  - Descripción de puestos anteriores, donde el entrevistador puede hacer cuestiones de anteriores trabajos o estudios realizados por el candidato, tanto responsabilidades, como razones por su marcha, entre otros.
  - Incidentes críticos, donde el candidato debe responder a preguntas relativas a como el candidato resolvería (o resolvió) problemas planteados por el entrevistador, y que así este pueda ver como el usuario procesa una situación crítica.
  - Clarificaciones. En este apartado, tanto el entrevistador como el candidato realizan preguntas acerca de detalles que no hayan quedado definidos durante el resto de la entrevista.
- **Cierre y despedida**. El entrevistador ha conseguido los datos necesarios para realizar su valoración, y con ello termina la entrevista. Normalmente, esta valoración no suele ser comunicada al candidato.

La estructura fluida de las entrevistas hace que los candidatos no puedan prepararse para estas hasta el momento en el que se realizan, y que, especialmente, es candidatos jóvenes, la inexperiencia a la hora de responder y los nervios puedan ser elementos negativos que, potencialmente, pueda terminar con una valoración negativa por parte de la empresa que perjudique al candidato [73].

## 2.4. Aplicaciones similares

El objetivo de este Trabajo de Fin de Grado, como se definió en el anterior apartado de Objetivos (apartado 1.2), es crear una aplicación, basada en Android, que permita

realizar la simulación de entrevistas de trabajo utilizando un sistema multimodal basado en un sistema de diálogo hablado, que pueda ofrecer *feedback* a los usuarios sobre su rendimiento en dichas simulaciones. Puesto que el sistema de Android ofrece a los desarrolladores la posibilidad de distribuir sus aplicaciones en la tienda integrada en el sistema operativo (Google Play Store) para que cualquier usuario con un dispositivo compatible pueda descargarla y utilizarla, en Android hay un alto número de aplicaciones disponibles para los usuarios, y como muestra de ello se puede destacar que sólo en 2017 los desarrolladores lanzaron más de un millón y medio de aplicaciones nuevas en la Play Store [74]. Es por ello que hay aplicaciones para muchas actividades, y que es posible encontrar aplicaciones basadas en el mercado de la simulación de entrevistas de trabajo. Para poder tener una visión global de la situación de este nicho del mercado, en este apartado se detallan, brevemente, una selección de algunas aplicaciones de carácter similar a la propuesta en este proyecto, para ver las funcionalidades que contienen en comparación a la idea de este Trabajo de Fin de Grado.

La primera aplicación estudiada es **Interview Question and Answers** [75], una aplicación gratuita y con más de un millón de descargas que, en su descripción de la tienda, especifica que tiene más de 500 preguntas disponibles para los usuarios, además de consejos y ejemplos de curriculums vitae disponibles para los usuarios que los necesiten. En la Figura 2.16, se pueden ver tres ejemplos de pantallas básicas de la aplicación.

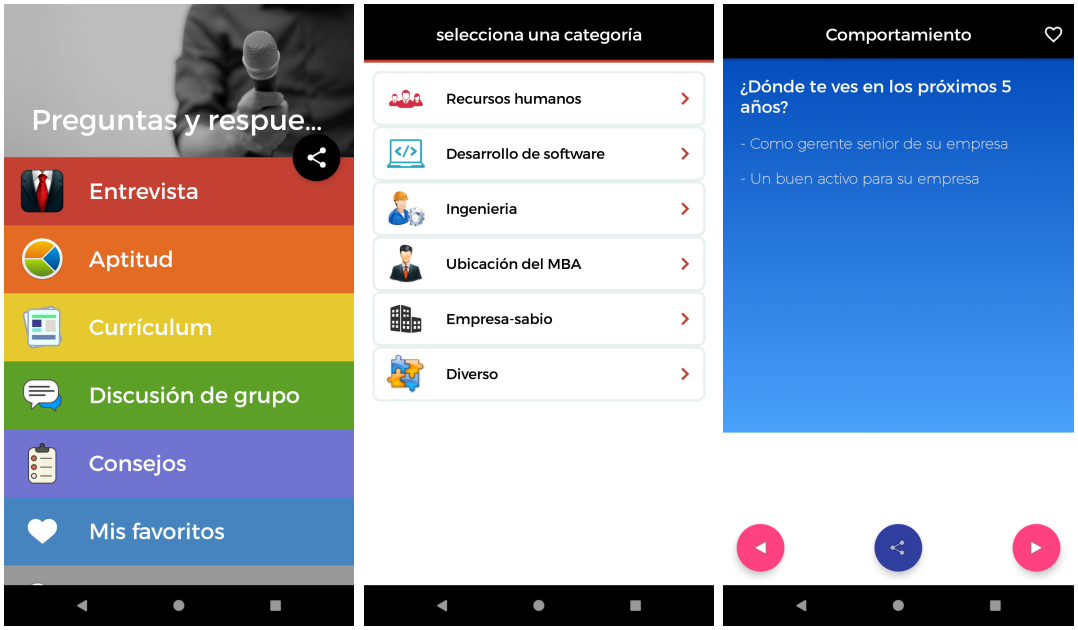


Figura 2.16: Pantallas de Interview Question and Answers

Probando la aplicación hay varios detalles a destacar. La sección de Entrevista no realiza una simulación de una entrevista, la idea de esta sección es que el usuario define el tipo de entrevista a la que, potencialmente, se sometería (por ejemplo, de ingeniería electrónica), y la aplicación le presenta una gran lista de potenciales preguntas que podrían ser formuladas en una entrevista real de ese tipo. El usuario puede entrar en cualquiera de estas preguntas, y ver una respuesta estándar que podría ser adecuada

para dicha pregunta. La interacción con la aplicación, en este punto, se define únicamente por la capacidad de leer respuestas que podrían ayudar al usuario en caso de tener que responder en algún momento a esas preguntas en una entrevista de trabajo real, y el usuario no introduce ningún tipo de información a la aplicación. Por otro lado, la sección de Aptitud es más interactiva, puesto que permite al usuario realizar pruebas genéricas de lógica y matemáticas en un formato test. Estas pruebas son cortas y de varios temas (series numéricas, razonamiento lógico y lingüístico, etc.), pero no ofrecen ningún tipo de sistema de retroalimentación al usuario sobre como ha sido su rendimiento en estas pruebas, simplemente se destaca la respuesta correcta al seleccionar una. El apartado de currículum es una página estática, donde el usuario tiene la opción de descargar a su dispositivo cinco plantillas en formato Word para tener referencia de varios estilos de currículum vitae y poder editarlos a su voluntad (fuera de la aplicación). La cuarta sección disponible es la sección de Discusión de grupo, la cual, de la misma manera que la sección Entrevista, ofrece al usuario la capacidad de seleccionar temas de conversación en los que quiere involucrarse (política, educación, temas de actualidad...), ver una lista de preguntas relacionadas con esos temas y posibles contestaciones con los que puede dar una buena imagen en caso de encontrarse hablando de esos temas. Finalmente, la última sección disponible es la de Consejos, donde los usuarios pueden encontrar consejos relacionados con varios temas del proceso de selección laboral en el que puedan estar, como normas de vestimenta, o cómo interactuar con el entrevistador, entre otros muchos temas.

En definitiva, esta aplicación podría considerarse una base de datos acerca de múltiples aspectos de entrevistas de trabajo disponible para cualquier usuario Android, pero la interacción con la aplicación es extremadamente limitada, y diseñada para únicamente ser leída. En comparación a este proyecto, esta aplicación carece de funcionalidades de interacción básicas que pudiesen simular una entrevista tal y como se quiere hacer aquí, pero la cantidad de preguntas y consejos disponibles ofrecen una fuente de recursos importante para los usuarios.

La siguiente aplicación encontrada se llama **Mock Interview** [76], y es de la misma compañía que la aplicación anterior (Placecom). La aplicación, de nuevo, es gratuita, pero ha sido descargada considerablemente menos veces que la anterior, únicamente por más de 50000 usuarios en comparación a la cifra superior al millón de la anterior aplicación. Viendo la descripción de la tienda, se observan diferencias clave respecto a la anterior, el objetivo de esta aplicación es hacer simulaciones de entrevistas de trabajo donde los usuarios puedan grabarse y evaluarse ellos mismos. Esta aplicación solo está disponible en inglés. En la Figura 2.17, se pueden ver ejemplos de las pantallas básicas de la aplicación.

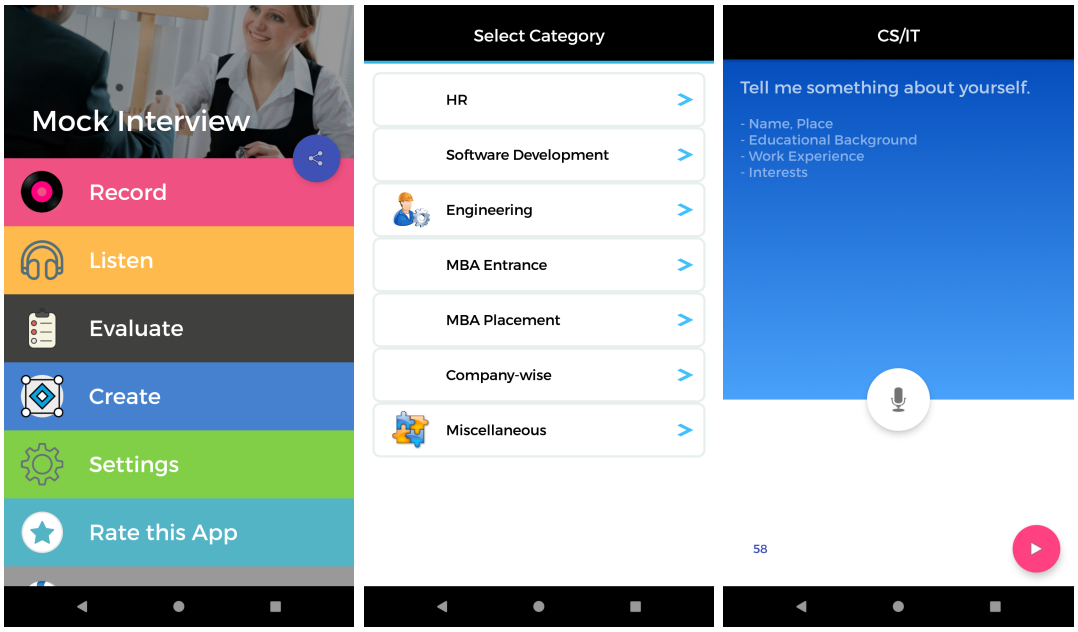


Figura 2.17: Pantallas de Mock Interview

Como se puede observar, la interfaz es prácticamente idéntica a la de la aplicación anterior. A pesar de ello, las funcionalidades de la aplicación difieren ligeramente, y es que el enfoque de esta aplicación es el de hacer una simulación de una entrevista de trabajo, donde el entrevistador realiza preguntas al usuario y este debe responderlas en menos de 60 segundos utilizando el micrófono del dispositivo. En la sección de *Record*, el usuario puede acceder a una lista de entrevistas para varios sectores laborales, y al entrar a una, la aplicación dicta al usuario una serie de preguntas (con voz sintetizada gracias a un módulo Text-To-Speech) que el usuario debe responder utilizando su voz. La aplicación graba un archivo de audio, sin hacer ningún tipo de reconocimiento de habla, que permite al usuario escuchar sus respuestas en la sección *Listen*. La aplicación no realiza ningún tipo de análisis a este archivo de audio, por lo que no evalúa al usuario su rendimiento, pero permite que este se evalúe a si mismo en la sección *Evaluate*, donde los usuarios pueden reproducir la entrevista entera y asignarse a si mismo una puntuación del 1 al 5. Esta evaluación es completamente subjetiva, y no es una representación real del rendimiento del usuario en la simulación evaluada. Por último, la aplicación permite, en la sección *Create*, generar entrevistas personalizadas, donde le usuario puede escoger de una lista de preguntas un conjunto de estas, darles un orden, y luego poder utilizarlas en futuras simulaciones.

Comparándola con este proyecto actual, esta aplicación ofrece el primer paso para conseguir una simulación de entrevistas de trabajo, que es crear un flujo de preguntas que el usuario pueda responder utilizando su dispositivo, y hay que destacar que el uso del Text-To-Speech en la aplicación ayuda a integrar al usuario en la experiencia. Sin embargo, Mock Interview no tiene ningún sistema de evaluación automática que ofrezca *feedback* al usuario sobre su rendimiento en estas entrevistas falsas, y la interacción con el usuario se reduce a ofrecerle la capacidad de grabar archivos de audio ofreciendo un contexto de una pregunta, funcionalidad que podría haberse integrado en la aplicación anterior y que podría haber aprovechado la gran base de datos de

preguntas y respuestas que contenía.

La siguiente aplicación se llama **Interview Simulator** [77]. Esta aplicación tiene una versión gratuita, pero también una versión de pago que desbloquea algunas funcionalidad disponibles en la aplicación, como simulaciones de mayor nivel o tutoriales en forma de video para alguno de los pasos del proceso de selección. El número de descargas es el más bajo de las tres aplicaciones vistas hasta ahora, con más de 5000 descargas, pero la descripción de la aplicación en la tienda indica que su proceso de simulación ha conseguido que se nombre en dos libros acerca de comunicación en un entorno profesional [78] [79]. Las múltiples pantallas de esta aplicación se pueden ver en la Figura 2.18:

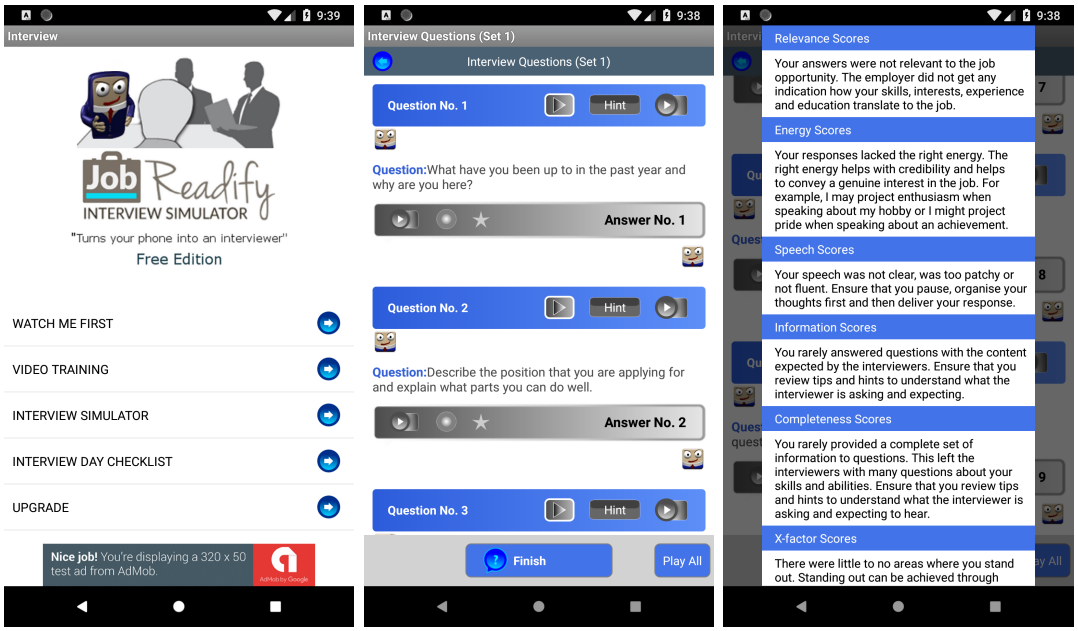


Figura 2.18: Pantallas de Interview Simulator

En esta aplicación, hay tres funcionalidades principales. La primera es la visualización de videotutoriales (*Video Training*), que permite a los usuarios ver una serie de videos, que cubren varios temas dentro del proceso de la selección laboral, y que utilizan animaciones y locución profesional para explicar consejos que puedan ser de ayuda a los usuarios en situaciones reales. La segunda sección principal es la más desarrollada, y es la simulación de entrevistas (*Interview Simulator*). En esta sección, los usuarios pueden elegir la dificultad y longitud de la entrevista a la que quieren enfrentarse, teniendo un rango desde 9 preguntas con un tiempo estimado de 20 minutos hasta entrevistas con 14 preguntas y un tiempo estimado de 45 minutos, y en las opciones de esta sección permite a los usuarios determinar si la entrada de información la harán por grabación de audio o por vídeo. Una vez comenzada la entrevista, el papel del entrevistador está locutado por actores reales, sin utilizar voz sintetizada, y se le ofrece al usuario la capacidad de leer o escuchar consejos, también locutados por parte de otro locutor, para responder correctamente a la pregunta. El usuario debe responder en inglés a las preguntas, puesto que el programa solo reconoce este lenguaje. Cuando la entrevista termina, se ofrece al usuario la posibilidad de escucharla

entera, y de recibir un informe de su rendimiento. Este informe contiene datos como el tono utilizado, la relevancia de las respuestas, la información recibida en la aplicación, etc. Subjetivamente, las pruebas realizadas en esta fase, no han dado los resultados esperados, ya que la aplicación no consiguió nunca reconocer que la respuesta pudiera tener relevancia o no, aunque el análisis de tonos era correcto. La conclusión es que el análisis realizado se basa en el archivo de audio generado en la grabación, y que, aunque realiza un análisis tonal y acústico, no consigue reconocer la información contenida dentro del mensaje de dicha grabación. Por último, la aplicación ofrece una serie de preparaciones para enfrentarse a una entrevista de trabajo (*Interview Day Checklist*), una lista que los usuarios pueden utilizar para asegurarse que cumplen ciertos consejos de cara a una entrevista real.

Esta aplicación ofrece las cualidades más interesantes de las vistas hasta el momento. En comparación a la aplicación propuesta por este proyecto, esta aplicación se acerca a la idea de simular una entrevista de trabajo y ofrecer retroalimentación al usuario creada automáticamente por el sistema, y se ofrecen varias maneras para interactuar con este (audio y video), cumpliendo requisitos para ser un sistema multimodal. Sin embargo, en las pruebas realizadas, el análisis no era completamente preciso, y, aunque su análisis sonoro era correcto, el de contenido no consiguió los resultados esperados, siendo esta una de las partes más importantes en una entrevista. También hay que destacar que, de cara a un público hispanohablante (o sin conocimientos de inglés), esta simulación no puede realizarse, ya que la aplicación solo funciona en inglés.

La última aplicación seleccionada es **Competencies** [80]. Esta aplicación es gratuita, pero gran parte de las funcionalidades están tras un servicio de suscripción. El número de descargas es de algo más de 5000 descargas, pero el bajo número de descargas se puede explicar en que esta es una aplicación basada en el uso de tecnología de realidad virtual (VR). Según la descripción, la utilización de VR coloca al usuario virtualmente dentro de una sala de entrevistas, donde modelos tridimensionales de los entrevistadores comienzan a hacer preguntas al usuario. En la Figura 2.19, se pueden ver algunas muestras de la representación virtual encontrada en esta aplicación, aunque con el formato original utilizado en dispositivos de realidad virtual:


Figura 2.19: Pantallas de Competencies [80]

Es conveniente destacar que prácticamente todas las funcionalidades están bloqueadas para usuarios de la versión gratuita, y eso incluye todas gran parte de las entrevistas, incluyendo la entrevista personalizada y el poder tener *feedback* al acabar una entrevista, por lo que el análisis de esta aplicación es limitado. Los modelos son completamente estáticos, y la voz está sintetizada virtualmente. Sin embargo, en la descripción de la aplicación se encuentra información de las funcionalidades esperadas en la versión de pago. El elemento más importante es el análisis de rendimiento de entrevistas, el cual se fija en tres elementos del rendimiento del usuario: el ritmo del usuario a la hora de responder, cuantas palabras de duda ha tenido y el contacto visual hecho a los agentes virtuales. El análisis no contempla ningún análisis relativo al contenido real del mensaje transmitido por el usuario, la cual es el elemento más importante en una entrevista, pero es interesante el uso de la tecnología en la que se soporta la aplicación (realidad virtual) para estudiar elementos como el contacto visual del usuario con las representaciones virtuales humanas ante las que se presenta, la cual es un elemento que se tiene en cuenta por parte de los entrevistadores para

comprender el lenguaje natural del usuario a la hora de responder a ciertas preguntas.

En comparación a la aplicación de este proyecto, Competencies ofrece ciertos elementos extra que solo podrían ser contemplados en una aplicación de VR, pero el resto de la interacción con la aplicación no ofrece ningún tipo de información relevante al usuario para enfrentarse a entrevistas de trabajo, al contrario de alternativas vistas anteriormente.

En definitiva, Interview Simulator ofrece la experiencia más cercana a la desarrollada en este Trabajo de Fin de Grado, pero hay elementos de ella que pueden ser pulidos, como el reconocimiento de información del texto, que pueden ser logrados con otras herramientas como se ha podido ver en apartados anteriores. Esta es la única aplicación encontrada que intenta realizar un análisis de retroalimentación ofrecido al usuario, el resto de aplicaciones disponibles en la Play Store de Android se basan en ser bases de datos acerca de elementos de entrevistas de trabajo, como preguntas o consejos, y no ofrecen ningún tipo de análisis interno sobre la respuesta del usuario, objetivo principal de este proyecto.

# Capítulo 3

# Diseño y desarrollo del sistema

## 3.1. Descripción de las funcionalidades básicas

Este proyecto, siendo una aplicación de Android, necesita de funcionalidades críticas que los usuarios puedan utilizar. La aplicación se basa en la idea de la simulación de entrevistas de trabajo, donde el usuario hace el papel de entrevistado, y los módulos de la aplicación hacen la labor de entrevistador. Teniendo en cuenta esto, las entrevistas, tal y como se ha podido ver en el estado del arte, se realizan, principalmente, en un entorno donde ambos participantes (entrevistado y entrevistador) se encuentran de manera presencial, y conversan acerca de distintos temas que el entrevistador considere relevantes para el desarrollo del proceso de selección.

Puesto que la aplicación quiere simular estas entrevistas presenciales, es esencial que la aplicación realice esta simulación utilizando, principalmente, la voz. La voz es uno de los elementos clave de comunicación de los humanos, y permite expresar tanto información como emociones a otras personas, que, por lo general, son capaces de interpretar estos datos y generar una respuesta. En una simulación de este estilo, donde participan dos personas, la comunicación se realiza de manera directa entre cada participante, y ambos reaccionan en orden a los comentarios del otro. En el contexto de una entrevista, es una conversación ponderada hacia un extremo, donde el entrevistado responde con el detalle que considere necesario a las preguntas que el entrevistador realice. Es por ello, que la aplicación debe simular, en concreto, a solo uno de los participantes: el entrevistador. El sistema deberá ser capaz de, principalmente, reconocer la voz del usuario (mediante un módulo *Speech-To-Text*) y recoger la información que contiene, realizar una lectura de esta información, evaluarla y responder al usuario con otras preguntas que continúen con el flujo de la entrevista. Siendo el reconocimiento de habla la vía principal de interacción con el usuario, la aplicación, para poder considerarse un sistema multimodal, debe permitir múltiples vías de interacción, y en esta aplicación los usuarios deben ser capaces de interactuar tanto con el habla como escribiendo las respuestas por teclado. Ofreciendo ambas posibilidades, los usuarios pueden interactuar mediante el canal más adecuado para ellos, tanto por temas de accesibilidad como de comodidad. Por otro lado, el papel del entrevistador debe estar representado en canales de salida, y también debe ser capaz de contactar con el usuario con varios canales de interacción. Es por ello que, mediante la aplicación, el sistema se comunicará con el usuario utilizando tanto texto como voz sintetizada (gracias a

un módulo *Text-To-Speech*), siendo esta salida un espejo de la entrada del usuario, lo que ayuda a que la representación de ambos extremos de la aplicación se comuniquen utilizando las mismas herramientas y canales. Además de plantear nuevas preguntas al usuario, el sistema deber ser capaz de realizar un análisis de las respuestas del usuario. El sistema hará uso de tecnologías basadas en el procesamiento de lenguaje natural, como el reconocimiento de expresiones clave y la minería de texto, para realizar un estudio de la entrada propuesta por el usuario, y así conseguir una evaluación del rendimiento del usuario a la hora de responder a las preguntas planteadas. Finalmente, esta evaluación debe ser mostrada al usuario, para tener referencia de su rendimiento y así poder mejorar en futuras simulaciones.

Una vez definido esto, se puede ver que existe una funcionalidad principal: **realizar la simulación de una entrevista de trabajo presencial**, donde el entrevistado será el usuario y el entrevistador será el sistema representado por la propia aplicación. Aunque el usuario interactuará con una aplicación de Android, esta aplicación únicamente será la interfaz del usuario con el sistema de análisis situado debajo, que estará compuesto de un agente conversacional basado en procesamiento del lenguaje natural (que mantenga el flujo de la conversación con el usuario reconociendo sus respuestas, obteniendo su contexto y los términos clave insertados en estas, y generar una nueva pregunta que realizar al usuario) y sistemas de análisis de sentimientos (que harán un análisis de la información introducida para conseguir encontrar el tono y la intención del usuario a la hora de formularla). Con esto, se puede ver que este proyecto tiene tres elementos clave diferenciados: un agente conversacional, un conjunto de sistemas de análisis de sentimientos y una aplicación que engloba la interacción del usuario con estos sistemas.

Para lograr este objetivo principal, es necesario que la **aplicación Android** realice las siguientes funcionalidades, muchas de ellas sucediendo en la lógica del programa, y que no serán visibles al usuario:

- **Mostrar preguntas al usuario**. La aplicación debe, desde el primer momento, ofrecer al usuario la posibilidad de conocer las nuevas preguntas generadas por el agente conversacional para que este pueda responderlas. Estas preguntas serán mostradas por pantalla en forma de texto y reproducidas por el altavoz del dispositivo utilizando un módulo de *Text-To-Speech* que sintetice una voz que pueda dictar las preguntas recibidas.

- **Manejar la introducción de información por parte del usuario**. La aplicación debe ser capaz de contemplar dos canales de entrada de información por parte del usuario: el habla (que debe ser transformado a texto mediante un módulo *Speech-To-Text*) y la escritura por teclado (donde el usuario pueda utilizar un teclado virtual o físico para introducir las respuestas que encuentre convenientes).

- **Conexión con los sistemas de evaluación**. En esta fase, la aplicación enviará la información introducida por el usuario a los sistemas de análisis (el agente conversacional y los sistemas de análisis de sentimientos) y recibirá la respuesta de ambos. En el caso del sistema de diálogo, la aplicación, además de otros datos,

recibirá la próxima pregunta que debe ser planteada al usuario (volviendo a la primera funcionalidad descrita).

- **Evaluación de resultados**. Dados los resultados de los sistemas de evaluación anteriores, la aplicación hará un cálculo con estos que representará el rendimiento del usuario al responder a esta pregunta actual.

- **Generar un informe final**. Al llegar al final de la entrevista, evento señalado por el agente conversacional, la aplicación debe ser capaz de recoger las evaluaciones realizadas a todas las respuestas del usuario, y mostrarlas en la pantalla al usuario como un informe final para que pueda tener constancia de este informe y, así, ofrecer *feedback* al usuario sobre su rendimiento en la simulación actual.

Por otro lado, los sistemas de análisis integrados en la aplicación también deben realizar funcionalidades básicas y necesarias para que el proyecto pueda realizar su objetivo principal correctamente. El primero en ser desglosado será el servicio proveedor del **agente conversacional**. Algunas de estas funcionalidades descritas a continuación vienen integradas dentro de las APIs de conexión que utilizan para contactar con otros servicios, o de la propia lógica de los servicios de procesamiento de lenguaje natural utilizados, pero otras deben ser definidas por el desarrollador para conseguir resultados precisos y útiles para la utilización de la aplicación principal por parte del usuario. Por parte del agente conversacional, dicho sistema debe realizar las siguientes fases para asegurar la correcta integración dentro del proyecto:

- **Personalización del modelo**. El desarrollador encargado de ajustar el modelo interno del agente conversacional debe tener a su disposición la capacidad de poder crear las posibles intenciones (con sus frases de entrenamiento y las potenciales respuestas) y entidades que puede recibir el agente en su entrada, y así ajustar el funcionamiento del agente para el correcto funcionamiento del sistema. Esto se debe realizar cada vez que se quiera introducir una nueva entrevista dentro del sistema.

- **Reconocimiento de la intención del usuario**. El agente debe ser capaz de reconocer el significado de la información recibida como entrada por parte del usuario, y saber identificar a que tipo de patrón de su modelo interno se refiere.

- **Reconocimiento de entidades**. Una vez reconocida la intención de la entrada recibida, el agente deberá acceder a su diccionario de entidades y reconocer las potenciales palabras clave encontradas dentro de la información recibida. Estas entidades buscan definir la relevancia de la respuesta respecto a la pregunta formulada.

- **Generación de preguntas**. Definido en el modelo de la primera fase, el agente conversacional debe saber cuál es la siguiente pregunta que va a formular al usuario, y estas pueden formularse de varias maneras, según sea definido en la personalización del modelo.

- **Envío de resultados a la aplicación**. Por último, el agente conversacional enviará toda la información recuperada en las anteriores fases (intención, entidades

y la nueva pregunta a formular al usuario) para que la aplicación la administre localmente. Esta información puede ser enviada en varios formatos, siendo el más común en servicios similares el formato JSON.

Por último, el último módulo principal del sistema es un **conjunto de servicios de análisis de sentimientos** que proporcionará el tono y sentimiento de la respuesta del usuario. Este conjunto de servicios deben ser accedidos desde la aplicación por separado, y la aplicación, como se ha definido anteriormente, debe juntar todos los resultados obtenidos para conseguir un resultado final. Las funcionalidades necesarias para la correcta integración de estos servicios en el proyecto son las siguientes:

- **Generación de la polaridad global**. Los servicios de análisis de sentimientos deben ser capaces de, dada una entrada de texto plano, reconocer el sentimiento y tono del texto en todo su conjunto, y no únicamente de cada uno de sus términos. Pueden tener capacidades extra, como análisis de subjetividad, elementos del lenguaje como la ironía, o un ratio de confianza, que puedan complementar el análisis y, por ello, obtener resultados más precisos que puedan ser utilizados en la aplicación Android.
- **Envío de resultados a la aplicación**. Los servicios deben ser capaces de comunicar los resultados obtenidos de vuelta a la apliación, conteniendo en un formato JSON todos los parámetros descubiertos en el análisis (principalmente, la polaridad global del texto, pero también los elementos opcionales que puedan encontrar en su análisis)

En los siguientes apartados se desglosará el diseño y desarrollo de cada uno de estos grandes módulos del proyecto.

## 3.2. Servicios de análisis de sentimientos

Tal y como fue explicado en apartados anteriores, los servicios de análisis de sentimientos son una parte importante del análisis realizado por este proyecto, y busca ofrecer al sistema una representación subjetiva del texto, una muestra de la intención contenida dentro del texto que otros sistemas de análisis no podrían recopilar. Estos servicios de análisis de sentimientos deben cumplir los requisitos recopilados en al apartado 3.1.1, y deben ser flexibles para poder ajustar su análisis a los requisitos necesarios para el correcto funcionamiento de este proyecto.

En el anterior apartado, en la descripción general del sistema, se habla de la utilización de un conjunto de servicios de análisis de sentimientos, en vez de un solo servicio. Es el caso encontrado en el desarrollo de este Trabajo de Fin de Grado, ya que este proyecto hace uso de varios servicios de análisis de sentimientos de manera paralela, que ofrecen la posibilidad de obtener distintas conclusiones en los análisis realizados y así tener la capacidad de tener una imagen global del sentimiento del texto que, realizando el análisis en único servicio, podría verse comprometido por la posibilidad de que el servicio utilizado no tenga un modelo lo suficientemente flexible para tener la correcta comprensión del texto necesaria para un proyecto similar a este. La necesidad

de utilizar varios análisis de sentimientos precisos en este proyecto se debe a la necesidad de encontrar la capacidad de simular, lo más cercano posible, a un entrevistador humano que, gracias a su entendimiento natural del lenguaje, es capaz de entender la información que el candidato le presenta en una entrevista de trabajo, y no sólo utilizando el significado de las frases expresadas por este, si no con un análisis subjetivo del resultado final que el candidato formula, que abarca desde el entendimiento del tono y expresiones utilizadas, hasta la propia formulación sintáctica de las frases que pueden llevar connotaciones secundarias no reflejadas en un análisis puramente semántico. Como se puede ver en el Estado del arte, hay un alto número de servicios de análisis de sentimientos disponibles en Internet para ser utilizados por desarrolladores y analistas que quieren utilizar el *sentiment analysis* dentro de sus proyectos, como es el caso actual, y la posibilidad de utilizar el hardware encontrado actualmente en dispositivos Android (como pueden ser *smartphones*), que son capaces de realizar una multitud de tareas al mismo tiempo gracias a procesadores multinúcleo, permiten a los desarrolladores implementar la integración de varios de estos servicios al mismo tiempo sin reducir drásticamente el rendimiento del dispositivo y de la aplicación. El principal problema reside en la fragmentación de usuarios encontrado en el sistema Android. Android, al ser un sistema operativo multidispositivo, puede ser utilizado en dispositivos con grandes variaciones en el hardware, y es posible que implementaciones con una necesidad más alta de recursos funcionen peor en dispositivos con un hardware antiguo o de gama baja que en dispositivos nuevos o de gama alta. Es por ello que la decisión de limitar el número de servicios de análisis de sentimientos integrados en la aplicación (aunque dicha integración únicamente se realice mediante conexión a Internet) fue decidida pronto en el desarrollo del proyecto, para así no limitar la distribución de la aplicación en múltiples sistemas Android y que esta pudiese tener un rendimiento adecuado en múltiples dispositivos.

Sin embargo, dada la necesidad de un análisis preciso por las razones comentadas anteriormente, este límite se colocó en tres integraciones a servicios de *sentiment analysis*. Estos tres servicios trabajan paralelamente cuando el usuario genera una respuesta, y su análisis es independiente uno del otro, para luego sacar conclusiones dentro de la lógica de la propia aplicación. Para la elección de los servicios utilizados para formar este conjunto, fueron escogidos servicios cuyos modelos se basasen en técnicas de aprendizaje automático. La razón para esta elección reside en la flexibilidad en el ajuste automático de los modelos, que son capaces de aprovechar información proveniente no solo del proyecto actual, si no que son capaces de crear modelos más grandes aprendiendo de las integraciones realizadas por otros desarrolladores también, ofreciendo un amplio rango de comprensión del lenguaje respecto a modelos puramente estadísticos y basados en reglas. Otra ventaja sobre estos modelos es que, si el servicio tiene capacidad de crear diccionarios o polaridades personalizadas, el servicio de análisis podrá reconocer más fácilmente términos no descritos explicitamente en dichos diccionarios, pero que hacen referencia a los mismos conceptos o intenciones, y así tener mayor precisión a la hora de reconocer la polaridad sentimental del texto introducido. Además de estos factores de carácter tecnológico y funcional, los servicios escogidos ofrecen la capacidad de utilizar planes gratuitos que contienen todas las funcionalidades encontradas en planes de precio mayor y con una cantidad considerable de peticiones que, de cara al desarrollo de este proyecto, ayuda para determinar la

fiabilidad y la eficacia de dichos servicios sin la necesidad de pagar una cuota mensual para realizar el proceso de desarrollo correctamente.

Los tres servicios implementados en el proyecto son los tres estudiados en el Estado del arte: **MeaningCloud**, **Watson Natural Language Understanding** (Watson NLU) y **Aylien**. Estos tres servicios han sido escogidos por las razones concretadas anteriormente, y porque el análisis realizado en el Estado del arte sobre estos servicios ofrece la capacidad de conocer sus características y funcionalidades antes de intentar integrarlos en el proyecto. En los siguientes párrafos, se procederá a realizar una explicación del diseño de los modelos realizados en cada uno de los servicios, y los ajustes necesarios para que los resultados obtenidos sean óptimos.

### 3.2.1. MeaningCloud

Comenzando por MeaningCloud, la utilización de su API de *sentiment analysis* necesita de la generación de una clave API única que deberá ser utilizada por la aplicación a la hora de realizar la conexión para enviar y recibir información de esta. Esta API es única para la aplicación, y no puede ser utilizada por más aplicaciones. MeaningCloud tiene todos sus servicios disponibles para cualquier cuenta e integración que el desarrollador quiera realizar, menos el servicio de categorización profunda *Deep Categorization*, que ofrece la posibilidad de realizar un análisis temático del texto y, así, revelar la categoría a la que pertenece el texto. La relevante para este proyecto es la API de *sentiment analysis*, que conjunta algunos otros servicios de MeaningCloud para ofrecer una respuesta rica en detalles.

MeaningCloud utiliza peticiones HTTP para comunicarse con otros servicios. Cuando una aplicación o otro servicio quiere transmitir información a MeaningCloud, debe hacerlo mediante una petición HTTP de tipo POST, que apunta a la dirección descubierta de la API y que, en el cuerpo de la petición, envie un objeto de tipo JSON con la información requerida por el servicio. Este JSON puede ser muy completo, llegando a poder detallar un marco de opciones del análisis muy amplio, como restricciones para palabras desconocidas o un indicador de la reliabilidad del texto a la hora de ser formulado (faltas ortográficas, erratas...) para ajustar la severidad del análisis y no tener tanto en cuenta estos errores. En la Tabla 3.1, se especifica la estructura de campos obligatorios de dicho objeto JSON para una correcta comunicación con la API de MeaningCloud:

| Clave | Descripción |
|---|---|
| key | La clave de acceso a la API de la aplicación. |
| language | El idioma a utilizar para el análisis de sentimientos. Permite o bien reconocerlo automáticamente (auto), o bien definir manualmente el uso de inglés, español, francés, italiano, portugués o catalán (en, es, fr, it, pt o ca, respectivamente). |
| txt/doc/url | O bien el texto plano que la aplicación manda para analizar, el documento con texto a analizar, o la URL con el texto a analizar. Estas claves son exclusivas, solo puede haber una de ellas en el JSON, no se puede utilizar una combinación. |

Tabla 3.1: Estructura del JSON de petición de MeaningCloud [81]

Sin embargo, hay dos campos optativos que son muy útiles para ajustar el análisis para el correcto funcionamiento de cualquier proyecto. Estos son *model* y *ud*, que, respectivamente, representan la elección del modelo personalizado a utilizar y el diccionario creado por el desarrollador requerido. Tanto los modelos como los diccionarios personalizados se diseñan internamente dentro de la interfaz de MeaningCloud, y tienen un identificador único que puede ser utilizado en el JSON de petición para requerir la utilización de ambos en el análisis a realizar. Los modelos son definiciones de expresiones y términos, a los que también se les pueden asignar sinónimos para ampliar el rango de reconocimiento del modelo, y que pueden definirse con una polaridad creada de una manera subjetiva por el desarrollador del modelo. Al definir términos y expresiones relativas a un tema concreto, el modelo tiene una referencia superior del potencial significado de un texto de ese tema, lo que hace que el análisis de sentimientos de dicho texto esté más ajustado a las expectativas del desarrollador. Por otro lado, los diccionarios son también definiciones de términos, pero a los que se les asigna una ontología o concepto que los relaciona entre ellos (por ejemplo, Nissan entraría dentro de la ontología de vehiculos). Los modelos y los diccionarios se pueden relacionar entre si, utilizando la conexión entre los términos de los modelos con los del diccionario, y a su vez con la ontología, pudiendo sacar conclusiones acerca de la polaridad de los conceptos integrados en esa temática. En la Figura 3.1, se puede observar la interfaz básica de creación de términos para un modelo, donde se puede ver como la polaridad puede cambiarse y personalizarse.

| | | | | | | |
|---|---|---|---|---|---|---|
| | 5b13fe9da3129 | take forever | POLARITY | N | 0 | 2018-06-03 16:43:41 |
| | 5b13fe9daa571 | to share | POLARITY | NONE | 0 | 2018-06-03 16:43:41 |
| | 5b13fe9d9ea36 | too sweet | POLARITY | N | 0 | 2018-06-03 16:43:41 |
| | 5b13fe9d9fd37 | Turkish delight | POLARITY | NONE | 0 | 2018-06-03 16:43:41 |
| | 5b13fe9da6203 | water down | POLARITY | N | 0 | 2018-06-03 16:43:41 |

Figura 3.1: Fragmento de interfaz de creación de modelos

Los modelos personalizados son utilizados en el proyecto como referencia de la polaridad esperada en algunos términos concretos por lo que sería, en esta aplicación, el entrevistador. Puesto que el proyecto trata con respuestas del usuario que pueden ser muy genéricas, el modelo creado (llamado *interview*) para el análisis de sentimientos en MeaningCloud se basa en la descripción de expresiones genéricas que pueden tener un tono negativo en cualquier conversación entre un candidato a un puesto de trabajo y su entrevistador. Términos y expresiones que puedan señalizar dudas o desconocimiento (por ejemplo, ”no estoy seguro”, ”ni idea”, etc.) se definen en el modelo con una polaridad negativa, para que el sistema de análisis sea capaz de reconocer estos términos con una recepción algo negativa, al igual que lo haría un entrevistador real. Por el otro lado, algunos términos que representan conocimiento del usuario (los contrarios a los anteriores: ”estoy seguro”, ”estoy capacitado” se marcan con una polaridad más positiva. Los diccionarios, sin embargo, no han sido necesarios para mejorar el análisis, puesto que el análisis de conceptos es más relevante en el estudio interno realizado por el agente conversacional, más que en los servicios de análisis de sentimientos.

Por último, el servicio envía una respuesta a la petición HTTP realizada por la aplicación. Esta respuesta puede estar en dos formatos (la aplicación puede pedir que sea enviada tanto en formato JSON como en formato XML), pero en el desarrollo de este proyecto se ha decidido que, para mantener la simetría con la petición, esta respuesta llegue a la aplicación en formato JSON. Esta respuesta viene dada con un amplio número de parámetros, que componen todo el resultado del análisis de sentimientos realizado. En la Tabla 3.2, se especifican los campos encontrados en este JSON de respuesta más relevantes para el funcionamiento de este proyecto.

| Clave | Descripción |
|---|---|
| score_tag | La polaridad global del texto. Se representa con seis valores: uno en caso de que el tono del texto se considere objetivo (NONE) y cinco en caso de que se considere un tono subjetivo (P+, P, NEU, N y N+, que, en orden, van de más positivo a más negativo). |
| agreement | MeaningCloud, aparte de la polaridad global, realiza un estudio de cada uno de los fragmentos del texto. En caso de que la mayoría de fragmentos tengan la misma polaridad, se devuelve el valor AGREEMENT. En caso contrario, si hay un desacuerdo entre las polaridades, se devuelve el valor DISAGREEMENT. |
| subjectivity | Valor que representa si el texto es subjetivo (SUBJECTIVE) u objetivo (OBJECTIVE). |
| confidence | El ratio de confianza del servicio con el resultado del análisis. Un valor cercano al 0 indica desconfianza con el resultado obtenido, mientras que un resultado cercano al 1 indica acierto en el resultado. |
| sentence_list | Array de fragmentos del texto. Cada uno de estos fragmentos incluyen un análisis independiente que utiliza los mismos campos que el texto global. |

Tabla 3.2: Estructura del JSON de respuesta de MeaningCloud [82]

Una vez recibido este objeto en la aplicación, esta debe recoger estos valores y realizar un análisis interno que tenga, principalmente en cuenta, los valores de polaridad global (score_tag) y la confianza en el resultado (confidence) que será explicado en el futuro apartado de Aplicación. MeaningCloud no ofrece ningún SDK compatible con Android, por lo que la integración deberá ser realizada manualmente.

### 3.2.2. Watson NLU

El segundo servicio de análisis de sentimientos utilizado es Watson Natural Language Understanding, o Watson NLU. Este servicio, como se puede ver en el Estado del arte, es un servicio ofrecido por IBM dentro de su nube de servicios IBM Cloud. Para utilizar cualquier servicio de IBM Cloud, es necesaria una cuenta en su sistema, que puede ser generada gratuitamente. Para la utilización del servicio Watson NLU,

es necesaria la generación de una clave (compuesta por un usuario y contraseñas generados por el servicio) para acceder a la API, que es única para la aplicación que quiera utilizar el servicio y todas sus funcionalidades. Watson NLU, dada su integración en la plataforma IBM Cloud, permite la integración de servicios alternativos de la misma plataforma para crear un flujo de transmisión de información entre ellos, y que, así, ciertas funcionalidades presentes en otros servicios puedan complementar los análisis realizados en Watson NLU. Watson NLU, además de funcionalidades de análisis de sentimientos, tiene integrada la posibilidad de utilizar otros análisis, como pueden ser la categorización del texto o de conceptos determinados o el descubrimiento de la relación temática entre varios términos del texto. Además, Watson NLU, como se pudo ver en el Estado del arte, tiene dos tipos de análisis de sentimientos: uno básico, similar al encontrado en MeaningCloud, y otro más detallado, que determina, aparte de la polaridad, la emoción concreta del texto recibido, utilizando cinco emociones como base: felicidad (*joy*), tristeza (*sadness*), miedo (*fear*), asco (*disgust*) y enfado (*anger*). Sin embargo, este último servicio, a pesar de ofrecer un análisis más completo que el básico, de momento solo está disponible para textos en inglés, y no en castellano, que es el idioma utilizado actualmente en este proyecto. Es por esta razón que, para la integración de Watson NLU en la aplicación, se utiliza el análisis de sentimientos básico.

La conexión con el servicio Watson NLU se puede realizar de varias maneras. Al igual que MeaningCloud, Watson NLU permite utilizar peticiones HTTP estándar para comunicarse con el servicio. Estas peticiones se deben realizar utilizando una petición de tipo POST, cuyo cuerpo contenga un JSON con toda la información esencial para el correcto funcionamiento del servicio junto con la calve de la API generada por la instancia del servicio. Este JSON tiene una estructura algo distinta a la encontrada en MeaningCloud. Mientras que MeaningCloud tiene una dirección de peticiones dedicada a realizar funciones de *sentiment analysis*, Watson NLU tiene una dirección de API genérica, que es utilizada por todas las peticiones a cualquiera de las funcionalidades contenidas en el servicio, por lo que el JSON de petición debe determinar que tipo de análisis debe realizar. En la Tabla 3.3, se detalla los parámetros necesarios contenidos en el JSON para que el servicio realice una comunicación correcta con la aplicación de integración:

| Clave | Descripción |
|---|---|
| text/html/url | Estos parámetros son exclusivos uno respecto al otro, y no pueden ser utilizados juntos. Respectivamente, contienen el texto a analizar, código HTML que pueda ser leido por el servicio o una dirección URL que el servicio pueda acceder. |
| features | Las funcionalidades que el usuario desee utilizar. Aquí se confirma qué análisis desean ser realizados (en el caso del *sentiment analysis*, se utilizaría el valor sentiment) y los parámetros extra que quieran ser utilizados en cada uno de esos análisis. En el caso del análisis de sentimientos se puede determinar el parámetro targets, que se compone de una lista de palabras clave que el usuario quiera analizar individualmente aparte del análisis global del texto. |
| language | El lenguaje a utilizar en el reconocimiento del texto. Se permiten 10 idiomas, entre los que se encuentran el español (es) y el inglés (en). |

Tabla 3.3: Estructura del JSON de petición de Watson NLU [66]

A diferencia de MeaningCloud, Watson NLU no permite crear modelos o diccionarios personalizados que puedan ser integrados en el análisis, por lo que el desarrollador se ve obligado a utilizar el modelo por defecto, que utiliza sistemas de aprendizaje automático para adaptarse y aprender dentro de una misma instancia del servicio. Es por ello que una instancia que trabaja con textos basados en, por ejemplo, términos tecnológicos, acabará generando polaridades distintas que un texto basado en un tema financiero. Aunque este ajusto automático es interesante para el desarrollador, ya que reduce el desarrollo necesario para que el análisis sea certero, la falta de personalización puede ser un detrimento de la precisión del análisis de sentimientos realizado ante algunos temas que necesiten de definiciones explicitas antes de funcionar correctamente. En el apartado de la evaluación del sistema, se podrá observar si este aprendizaje totalmente automático, y la falta de personalización, afecta a los resultados esperados.

Utilizando la respuesta de la petición original, Watson NLU se comunica con la aplicación utilizando un objeto JSON que contiene los resultados obtenidos en los análisis requeridos por la petición. La estructura es diferente a la encontrada en la petición, y contiene un alto número de parámetros en caso de haber realizado varios análisis. En la Tabla 3.4, se definen los campos relevantes encontrados al realizar un análisis de sentimientos:

| Clave | Descripción |
|---|---|
| targets | Array de palabras claves definidas en la petición. Cada una de las palabras contienen los valores de score y label, que representan la polaridad (mostrada de -1 al 1, siendo estos respectivamente el valor más negativo y el más positivo) y una etiqueta textual de la polaridad (positive, neutral o negative.) |
| document | De la misma manera que el servicio puede realizar análisis a términos concretos, también realiza el análisis al documento global. Al igual que en targets, este parámetro contiene los valores de score y label, que representan la polaridad global del texto. |

Tabla 3.4: Estructura del JSON de respuesta de Watson NLU [66]

De nuevo, la aplicación integrará el uso de este servicio de manera paralela al del resto de servicios de análisis de sentimientos, y utilizando, principalmente, el score del documento global, deberá realizar un análisis del resultado obtenido por los tres servicios de *sentiment analysis*. La integración con la aplicación se realizará mediante el SDK de Watson NLU disponible para Android, que ofrece una interfaz para crear y leer estas conexiones HTTP de manera sencilla para el desarrollador de la aplicación.

### 3.2.3. Aylien

Por último, el último servicio de análisis de sentimientos integrado en el sistema es Aylien. Aylien, al igual que los otros dos servicios vistos, utiliza una clave para acceder a la API para el usuario, y otra para la aplicación donde se integra el servicio. Aylien, además de realizar análisis de sentimientos, es capaz de hacer análisis alternativos como la extracción de temáticas de textos o de entidades clave de este, al igual que los otros dos servicios. El *sentiment analysis* se puede realizar mediante dos métodos, cada uno enfocado a un elemento distinto de la entrada. El primer método es el *sentiment analysis* global, donde el servicio recoge el texto entero y realiza un análisis de sentimientos de todo él. Sin embargo, el segundo método es un análisis de sentimientos basado en aspectos del texto (*Aspect-Based Sentiment Analysis*), que, en vez de realizar un análisis de sentimientos del texto, el usuario puede requerir la polaridad de un concepto encontrado en el texto (por ejemplo, este texto tiene una polaridad relativa a los hoteles positiva). Actualmente, Aylien ofrece la posibilidad de realizar este análisis sobre los conceptos de hoteles, restaurantes, coches y aerolíneas, por lo que no es útil para este proyecto, y conlleva que la integración con este servicio utilice la funcionalidad de *sentiment analysis* normal.

Aylien, como gran parte de los servicios similares, utiliza comunicaciones HTTP para compartir información entre la aplicación y el propio servicio de Aylien. La aplicación debe realizar una petición POST enviando un objeto JSON con la información necesaria para realizar el análisis. En este caso, el JSON a formar es más simple que el necesario en los otros dos servicios, siendo necesarios solo tres parámetros (sin contar las claves de acceso a la API). En la Tabla 3.5, se especifican los elementos necesarios para componer un JSON correcto que Aylien pueda leer:

| **Clave** | **Descripción** |
|---|---|
| mode | Aquí se puede determinar el tipo de texto enviado. Puede ser de tipo tweet (textos cortos) o document (para textos más longevos). Dependiendo del modo utilizado, el análisis se ajusta para tener mejor precisión ante textos cortos o largos. |
| text/url | O bien el texto a analizar, o bien la URL de la página web cuyos contenidos desean ser analizados |
| language | El lenguaje a utilizar en el reconocimiento del texto. Se permiten tres idiomas: español(es), inglés (en) y alemán (de) |

Tabla 3.5: Estructura del JSON de petición de Aylien [83]

Igual que Watson NLU, y al contrario que MeaningCloud, Aylien no permite crear modelos o diccionarios personalizados. Utilizando un sistema similar al servicio de IBM, su modelo interno se ajusta a cada aplicación de manera automática, sin permitir ningún tipo de personalización por parte del desarrollador, lo que puede reducir la precisión del análisis.

Al igual que con la petición, Aylien utiliza una conexión HTTP para enviar su respuesta. Esta respuesta, igualmente, se genera dentro de un objeto JSON, que es considerablemente más pequeño que el encontrado en los JSON de respuesta de MeaningCloud y Watson NLU. En la Tabla 3.6, se detallan cada uno de los parámetros de este JSON de respuesta:

| Clave | Descripción |
|---|---|
| polarity | La polaridad global del texto. Se representa con dos valores: positive o negative. |
| subjectivity | Representación de la subjetividad del texto. Puede ser subjective o objective. |
| polarity_confidence | Confianza (representada con un valor del 0 al 1) que el sistema de análisis tiene con la polaridad obtenida. |
| subjectivity_confidence | Confianza (representada con un valor del 0 al 1) que el sistema de análisis tiene con la subjetividad obtenida. |

Tabla 3.6: Estructura del JSON de respuesta de Aylien [83]

Al igual que con los otros dos, la aplicación integrará (en este caso, gracias a un SDK de integración disponible para Android) este servicio, y analizará internamente todos los resultados como se podrá ver en la sección Aplicación.

## 3.3. Agente conversacional

El segundo gran módulo de este sistema de simulación de entrevistas de trabajo es el encargado de generar esta misma entrevista, y mantener una conversación con el usuario que pueda ser analizada tanto por el propio agente como por los servicios de análisis de sentimientos descritos en al apartado anterior. Este módulo es un agente conversacional, que será integrado en la aplicación para realizar un análisis semántico del texto, capaz de extraer conceptos clave de la información introducida y posteriormente mantener un flujo de conversación básico con el usuario de la aplicación.

La integración de este módulo en la aplicación es esencial para el correcto funcionamiento de esta. En el caso de este proyecto planteado, donde los usuarios responden a preguntas planteadas por un entrevistador virtual, es necesario que haya un sistema donde dichas preguntas sean formuladas de vuelta a la aplicación y que cualquier respuesta generada por el usuario sea estudiable en el contexto de dicha pregunta. Mientras que los sistemas de análisis de sentimientos hacen un análisis sintáctico del texto de las respuestas del usuario, obteniendo un tono y emoción encontrados en la formulación del texto recibido, y sin tener en cuenta la pregunta a la que están respondiendo, un agente conversacional es capaz de manejar el contexto de la información recibida, y es capaz de realizar un análisis semántico que tenga en cuenta este contexto y sepa relacionar los conceptos encontrados en la respuesta con los esperados al formular una pregunta, de una manera similar a la que lo haría un entrevistador humano con comprensión del lenguaje utilizado. Un agente conversacional, al poder concretar la relación entre una pregunta y una respuesta, es capaz de realizar un análisis de la relevancia de dicha respuesta ante dicha pregunta, elemento clave de la evaluación realizada por un entrevistador cuando plantea preguntas a un candidato. Haciendo

el símil entre una entrevista real y este sistema, un candidato, al igual que debe responder utilizando un tono positivo (analizado en este caso con servicios de *sentiment analysis*), debe responder con una respuesta relevante respecto a la duda planteada (análisis realizado por el agente conversacional), utilizando un entorno conversacional (que, en este proyecto, sería una interfaz representada en la aplicación Android).

En este sistema, se ha decidido utilizar el servicio Dialogflow, anteriormente visto en el Estado del arte, para poder desarrollar este agente. Dialogflow permite utilizar la gran mayoría de sus características en el plan gratuito que tienen disponible, y ofrece la flexibilidad necesaria para crear una conversación realista con un usuario. Este servicio se basa en la utilización de sistemas de aprendizaje automático integrados en la plataforma de servicios en la nube de Google, llamada Google Cloud. Estos sistemas de aprendizaje automático, de una manera similar a técnicas similares encontradas en MeaningCloud, Watson NLU o Aylien para *sentiment analysis*, son capaces de ajustar los modelos definidos por el desarrollador del agente para ajustarse a expresiones potencialmente no definidas explicitamente en dichos modelos, y así automáticamente aprender nuevos conceptos y expresiones mediante el uso del sistema por parte de los usuarios y los desarrolladores. Además, su integración en Google Cloud permite una mayor escalabilidad de cara a la posible expansión del sistema entre varios usuarios, que, en otros servicios, podrían bloquear el servidor y dejar sin servicio a todos ellos. También permite la integración de otros servicios de Google Cloud, como servicios de minería de texto o *Speech-To-Text* avanzados que podrían ayudar, potencialmente, al rendimiento de la aplicación y los resultados obtenidos por esta. Sin embargo, para este proyecto, se ha decidido que, para realizar el estudio de la fiabilidad de un sistema de simulación de entrevistas como el realizado actualmente, es mejor utilizar un plan gratuito que, sin tener estas carácterísticas extra, ofrece la posibilidad de definir la lógica de un agente completo sin restricciones.

Dialogflow permite crear varios agentes independientes, cada uno con sus carácterísticas y definiciones. Para este proyecto, solo es necesario crear un agente, que hará el papel de entrevistador. Este agente, en este proyecto, se llama, convenientemente, "Entrevistador". Los agentes, como se definió en el Estado del arte, están compuestos por dos elementos principales, que contienen la información necesaria para definir un modelo que pueda comprender los términos utilizados por los usuarios. Estos dos elementos principales son las intenciones *intents* y las entidades *entities*. A continuación se detallarán ambos elementos, dentro del contexto del modelo utilizado por el agente "Entrevistador", además de una definición de la conversación planeada entre el usuario y el agente, y su integración con la aplicación de Android.

### 3.3.1. Definición de conversación entre usuario y agente

Los agentes conversacionales, como pueden ser los creados por Dialogflow, hacen uso de técnicas de aprendizaje automático para ajustar sus resultados según los usuarios interactúan con él, y van ofreciendo información que puede ser utilizada por el agente y el desarrollador para ajustar su modelo interno. Sin embargo, para que un agente de Dialogflow entienda los diálogos recibidos por parte del usuario, es necesario

definir un flujo de conversación que contenga tanto intenciones como entidades relevantes para los mensajes que el agente espera recibir. Antes de definir estas *intents* y *entities* dentro de la lógica interna del agente de Dialogflow, es necesario realizar un plan del flujo que se requiere desarrollar en el sistema. En el caso de una entrevista de trabajo como la simulada, el flujo es sencillo: el entrevistador debe plantear preguntas a un candidato al puesto de trabajo y este debe responderlas, y así en un ciclo de preguntas y respuestas que dura hasta que el entrevistador lo considere necesario. Cuando un entrevistador escucha la respuesta de un candidato, recogerá elementos de dicha respuesta, que abarca desde el tono en el que fue formulada la respuesta hasta el significado propio de la respuesta y los términos y expresiones utilizados por el candidato. En este caso, el agente de Dialogflow debe plantear preguntas al usuario, y este responderlas de vuelta al agente, que realizará un análisis de los términos clave que el desarrollador desee encontrar en la respuesta perteneciente a cada pregunta. Una simple descripción del flujo de conversación planteado se puede observar en la Figura 3.2:

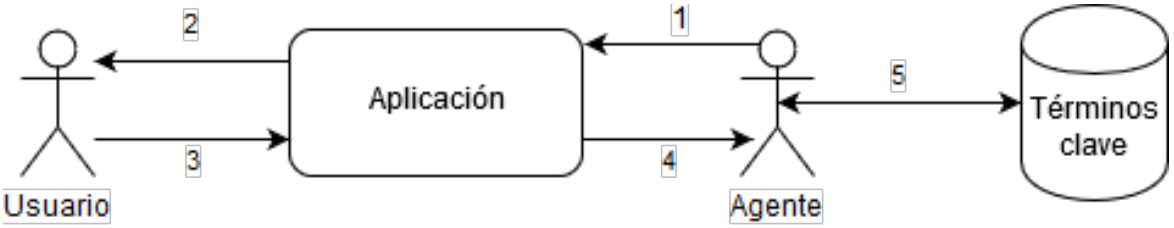


Figura 3.2: Flujo de conversación en el proyecto

En la figura anterior, se pueden observar varios pasos, definidos a continuación:

1. El agente de Dialogflow envía la información a la aplicación, entre lo que se encuentra una nueva pregunta que el servicio plantea al usuario.

2. El usuario recibe esta nueva pregunta, en formato texto o en formato sonoro, con voz sintetizada

3. El usuario, mediante el uso de la voz o escritura por teclado, genera una respuesta que introduce en la aplicación.

4. La aplicación manda al agente la respuesta introducida por el usuario

5. El agente consulta su modelo para reconocer los términos o expresiones clave que pueda haber en el texto recogido del usuario, y una vez reconocidos los guarda en la base de datos de la conversación (integrada dentro del propio agente de Dialogflow) y envía en la siguiente interacción esta información encontrada a la aplicación para que esta realice su análisis local del rendimiento del usuario.

Una vez completado en quinto paso, el agente vuelve a realizar el primer paso, con una nueva pregunta generada según estén definidas en su modelo y la información encontrada en el paso cinco.

Aunque se ha definido el flujo de la conversación, el propio plan de la entrevista no se ha definido en esta memoria. Los agentes conversacionales pueden realizar dos tipos de interacción con el usuario: diálogos abiertos o diálogos dirigidos. Los **diálogos abiertos** son los que se pueden encontrar en un sistema similar al asistente virtual Siri o Google Assistant, donde la conversación con el sistema se reduce a una sola consulta realizada por el usuario y a una sola respuesta realizada por el agente, lo que provoca que la interacción comience de nuevo al recibir la respuesta del agente. Estos diálogos se encuentran en un entorno libre, donde el agente está preparado para responder a múltiples intenciones de petición del usuario, sin esperar ningún patrón concreto según las interacciones anteriores. Es por ello que, por ejemplo, un agente como Siri puede responder a múltiples peticiones de un usuario, como, por ejemplo, el clima de hoy, y si el usuario realiza otra pregunta de un tema distinto olvida la interacción anterior y responde sin problemas a esta nueva petición. Un **diálogo dirigido** es el contrario: el agente introduce la interacción del usuario dentro de un camino definido de ejecución, donde espera que el usuario responda en el contexto del mismo hilo de conversación, y no salga de él. Si Siri funcionase así, el ejemplo de conversación anterior, donde el usuario pregunta por el clima, no podría terminar hasta que el agente lo decidiese, y si el usuario intentase realizar otra pregunta (por ejemplo, "¿Dónde puedo comer cerca?") el agente no reconocería la intención del usuario, e intentaría reconducir la conversación de nuevo al tema dirigido (en este caso, el agente respondería algo similar a "Esta pregunta no tiene que ver con el clima, no puedo responderte").

Los agentes conversacionales pueden funcionar de una manera híbrida, dependiendo de como esté definido el modelo interno, y podría utilizar, por ejemplo, un estilo de diálogo abierto para recibir una petición genérica del usuario, que se transformaría en un diálogo dirigido que, cuando terminase, volvería al diálogo abierto del principio para continuar con la interacción con el usuario. De hecho, ese es el caso del agente conversacional desarrollado para este proyecto, puesto que comienza con un diálogo relativamente abierto con el usuario para determinar qué entrevista quiere realizar (el usuario decide un número) y después el agente cierra la conversación a un diálogo dirigido, donde el agente plantea las preguntas de la entrevista en un orden previamente determinado por el desarrollador del modelo y obliga al usuario a responderlas en ese orden, de la misma manera que se haría en una entrevista real. Además de así poder simular una entrevista de una manera más realista, de esta manera se puede controlar el orden de la entrevista y el análisis a realizar a cada una de las respuestas generadas por el usuario, ya que conociendo el orden de las preguntas, comprender el contexto y significado de las respuestas es más sencillo. Si el sistema se hubiera diseñado con un diálogo abierto, el análisis no podría ser realizado correctamente, gracias a la tendencia de los candidatos a realizar respuestas genéricas para varias preguntas que, potencialmente, no tengan que ver con la pregunta realizada por el agente, evitando así la definición en el modelo de los elementos esperados, como expresiones clave.

En la definición de conversación de este proyecto, el usuario pide al agente que le introduzca dentro de una entrevista concreta, definida como una serie de preguntas realizadas en un orden concreto por el agente (estas preguntas fueron recogidas de distintas fuentes de preguntas comunes en entrevistas [84] [85]), para poder realizar su correcto estudio. Esta entrevista, dentro de Dialogflow, debe ser definida como una

colección de intenciones (*intents*) y entidades (*entities*), como se podrá ver en los próximos apartados.

### 3.3.2. Intenciones (*intents*)

Las intenciones, dentro de un servicio de agentes conversacionales como Dialogflow, es el elemento principal de un modelo de dichos agentes. Una intención se puede definir como la relación entre la información recibida por un usuario y la lógica interna del agente, que incluye todas las acciones que este debe realizar al recibir este tipo de información. Dialogflow, dentro de un agente, permite al desarrollador crear tantas intenciones como este vea necesario para que el agente pueda comprender las acciones necesarias que debe realizar en caso de reconocer la información recogida.

Tal y como se definió en el apartado anterior, el modelo del agente debe ser capaz de realizar una conversación con el usuario que sea cerrada cuando la entrevista comience. El agente planteará preguntas al usuario que este debe responder, y que, cuando responda, el agente pueda reconocer como una respuesta relacionada con la pregunta anteriormente planteada. El reconocimiento de las respuestas se realiza, dentro del agente de Dialogflow, creando tantos *intents* como preguntas realice el agente. Con una intención relacionada con una única pregunta, el agente es capaz de reconocer cada introducción de información como respuesta de la última pregunta realizada, y así el análisis puede ser definido con más precisión. En el caso concreto del agente utilizado para este proyecto (llamado "Entrevistador"), hay una única intención de entrada, que solo reconoce un solo elemento de entrada: un texto que, únicamente, contenga la palabra "*Start*". Esta palabra no es enviada por el usuario, si no por la propia aplicación automáticamente cuando el usuario accede a la pantalla de entrevistas. Con esto, el agente comienza a esperar la introducción de texto por parte de la aplicación, lo cual se refleja con una respuesta devuelta a la aplicación donde hace una petición al usuario de una confirmación de que quiere realizar una entrevista. En las Figuras 3.3 y 3.4, se puede ver dos fragmentos de la definición de esta primera intención dentro de Dialogflow, llamada internamente **Interview-Start**:

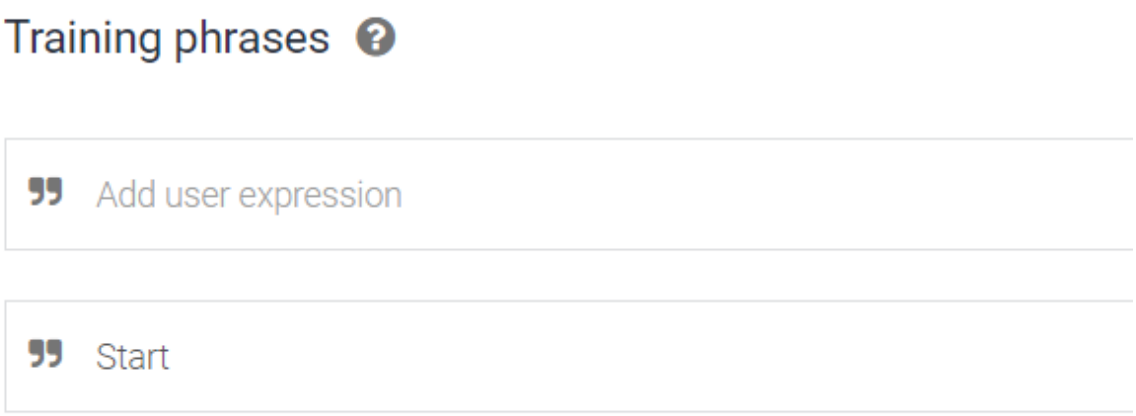


Figura 3.3: Frases de entrenamiento del *intent* Interview-Start

En la Figura 3.3 se puede ver la utilización de lo que en la interfaz de Dialogflow se llaman *training phrases*, en castellano frases de entrenamiento. Estas frases, tal y como fue descrito en el Estado del arte, son frases de ejemplo utilizadas por el agente para reconocer el *intent*. En la Figura 2.10 se puede ver un ejemplo más desarrollado, con varias frases de ejemplo, pero para esta intención la única frase de ejemplo reconocible es la palabra "*Start*". Un elemento interesante de Dialogflow es que hace uso de técnicas de aprendizaje automático para entrenar el reconocimiento de intenciones dentro de los agentes, y, en el caso de la *intent* Interview-Start, a pesar de únicamente utilizar una frase de ejemplo, podría utilizarse palabras como "Comenzar" o "Empieza" para hacerlo funcionar sin necesidad de explicitamente definirlas como frases de entrenamiento, ya que esta adaptación automática del modelo es capaz de interpretar el significado de una entrada haciendo uso de un análisis interno de minería de texto integrado en Dialogflow, invisible tanto para el usuario como para el desarrollador, y relacionar los resultados con las intenciones definidas por el desarrollador del modelo.

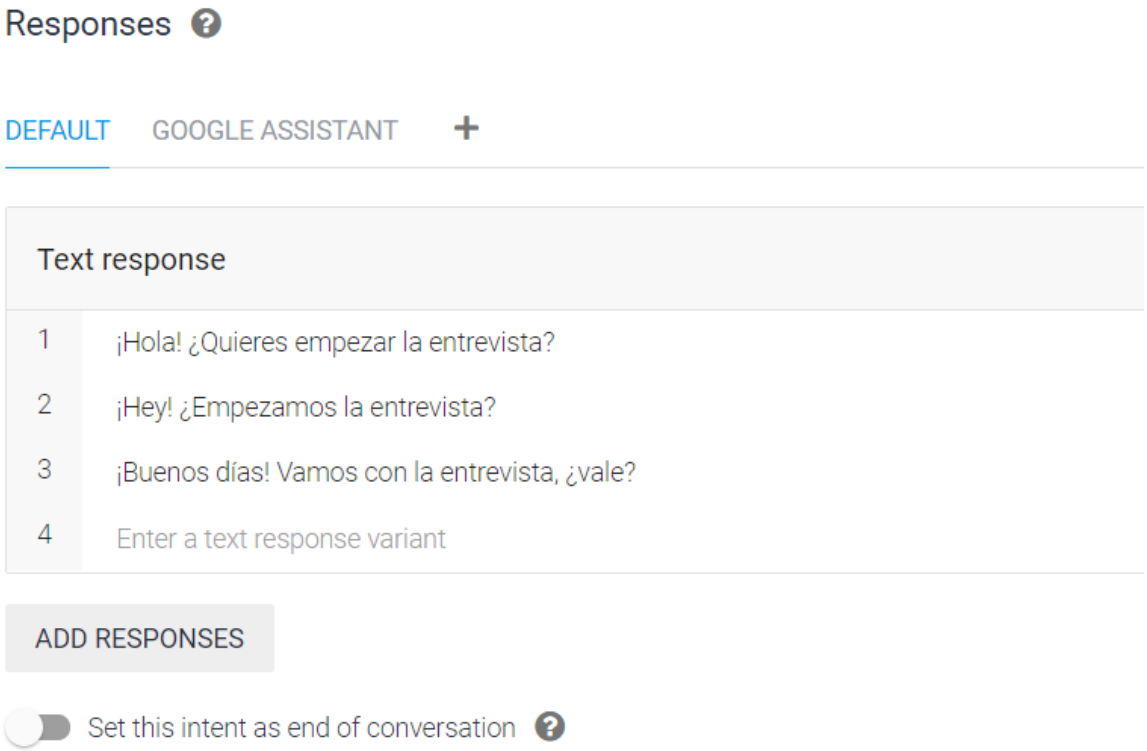


Figura 3.4: Respuestas posibles del *intent* Interview-Start

Por otro lado, en la Figura 3.4, se contienen las respuestas *responses* que el agente realizará al encontrarse en esta intención en concreto. Actualmente, como se puede ver, las respuestas de esta intención Interview-Start guían al usuario a una misma dirección, que es plantearle la posibilidad de realizar una entrevista. Aunque el término "Respuestas" puede llevar a pensar lo contrario, la respuesta del agente, en este proyecto, en realidad conlleva las preguntas de la entrevista que quiere realizar al usuario. Es por ello que, aunque dentro del marco teórico de la comunicación entre dos sistemas el concepto sea tal y como define Dialogflow (respuestas ante una petición realizada por el usuario), dentro de este proyecto en concreto es la definición contraria: estas "respuestas" en realidad son las preguntas que el entrevistador plantea al usuario. En esta sección, se pueden definir una multitud de potenciales respuestas que el agente devolverá a la aplicación, y el agente escogerá, al azar, una de ellas para enviarla de vuelta a la aplicación.

Una vez iniciado el agente, el usuario debe responder a la primera pregunta planteada por este, donde debe decidir si quiere realizar la entrevista o no. Este es el comienzo de la interacción entre el usuario y el agente. El usuario, por supuesto, puede responder afirmativamente o negativamente, y esto implica dos nuevas intenciones: una donde se recoge la negación (llamada Interview-Start-No), que termina la comunicación con la aplicación en ese momento y se lo indica al usuario con un mensaje, y otro que refleja la afirmación del usuario (Interview-Start-Yes). En este segundo caso, en donde el usuario confirma que quiere realizar una entrevista, se pasa a otra fase del diálogo

abierto donde la aplicación realiza la petición de un número al usuario. Este número representa la entrevista que el usuario desea realizar, y una vez confirmado el número el diálogo se convierte en dirigido, y comienza la entrevista. En las Figuras 3.5 y 3.6, se concretan las frases de entrenamiento y respuestas del comienzo de la entrevista número 1 (cuya *intent* es llamada **Interview 1.1**, refiriéndose a la entrevista 1, pregunta 1), aunque hay que destacar que todas las posibles entrevistas siguen un modelo similar (la intención para reconocer el número 2, por ejemplo, se llamada Interview 2.1, y así correspondientemente, y el modelo de frases de entrenamiento y respuestas también es similar).

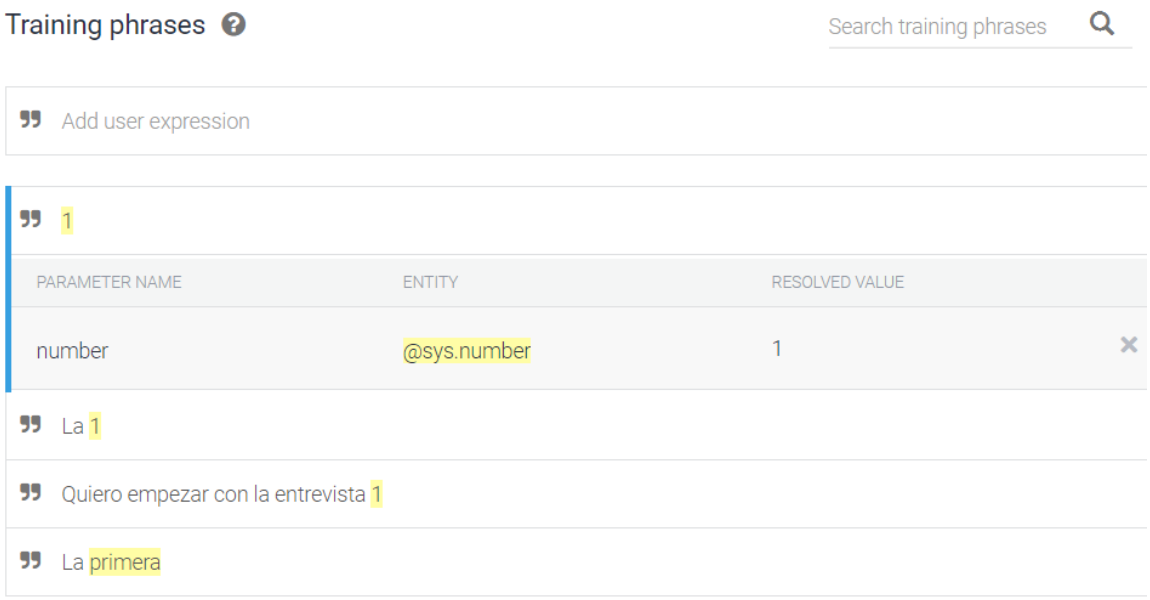


Figura 3.5: Frases de entrenamiento del *intent* Interview 1.1

Empezando por las frases de entrenamiento de la Figura 3.5, se puede ver una importante diferencia respecto a la anterior definición de *intent* vista, la de Interview-Start, y es que, aparte de definir más formaciones lingüísticas como ejemplo de frase de entrenamiento (utilizando números cardinales además de cifras), esta es la primera intención del modelo que hace uso del reconocimiento de entidades incorporado en Dialogflow. En este caso, se puede observar como el color amarillo representa a la variable *number*, que contiene la respuesta del usuario a la pregunta anterior (el número de la entrevista a realizar). Esta variable corresponde al tipo de entidad @sys.number, que, tal y como dice su prefijo, es uno de los tipos de entidad por defecto contenidos dentro de Dialogflow, y que pueden ser utilizados libremente por los desarrolladores de agentes para reconocer expresiones y términos utilizados genéricamente, como números, ciudades, horas o elementos básicos utilizados en el día a día como conceptos del clima, de viajes, etc. En el caso de la definición de estas frases de entrenamiento, la propia interfaz de Dialogflow señaló automáticamente estos elementos numéricos, pero una posibilidad presente para el desarrollador es señalar manualmente conceptos y determinar que tipo de variable y entidad representan.

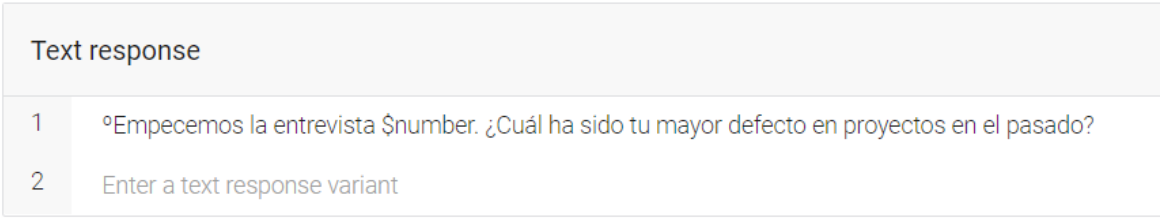


Figura 3.6: Respuestas posibles del *intent* Interview 1.1

Por otro lado, en la Figura 3.6, se puede observar la única respuesta utilizada por esta intención, y la utilización de la variable definida anteriormente. Para asegurar al usuario de que ha escogido correctamente la entrevista deseada, se replica la utilización del número recogido anteriormente en la respuesta, utilizando la variable $number. Utilizando esta variable, en este caso, la respuesta podrá asegurar que la entrevista seleccionada es la número 1. El otro componente de esta respuesta enviada a la aplicación es la primera pregunta de la entrevista. Como se especificó anteriormente, a partir de este momento, el diálogo con el usuario es dirigido completamente por el agente, y solo permite un camino con el que el usuario puede interactuar. El desarrollador define, en estas respuestas, el orden de la entrevista, y en este caso se puede ver como el agente no tiene más remedio que generar la primera pregunta utilizando esta base definida anteriormente. Por último, destacar que el carácter º encontrado al comienzo de la respuesta es un indicador de comienzo de la entrevista que la aplicación debe procesar para saber el punto en el que la interacción con el agente empieza el proceso de la entrevista.

Una vez terminado el procesamiento de este *intent*, el agente transforma el reconocimiento de intenciones a un estilo basado en la utilización de contextos (que son capaces de relacionar varios *intents* entre sí) para cerrar el diálogo posible a un solo camino definido por la entrevista. En la Figura 3.7, se puede observar visualmente como es esta relación entre intenciones dentro de la interfaz de Dialogflow:

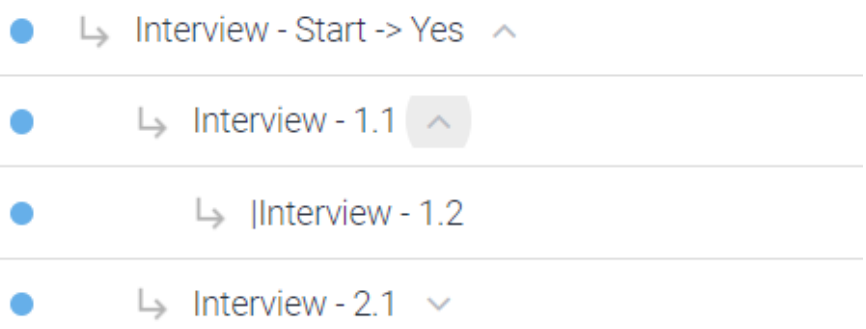


Figura 3.7: Relación entre *intents* en Dialogflow

Como se puede ver, las relaciones contextuales entre intenciones permiten generar un camino en el diálogo, y así crear un sistema de diálogo cerrado que pueda ser controlado por el agente correctamente. Los contextos no solo son capaces de generar

la relación entre los *intents*, si no que también son capaces de recopilar las diferentes variables encontradas en el reconocimiento de una intención y ofrecerlas a cualquier futura intención que se encuentre dentro del mismo contexto.

En este punto del proceso, la interacción entre el agente y la aplicación se encuentra en un bucle como el encontrado en la Figura 3.2. El agente plantea preguntas al usuario, que este responde, lo que provoca que el agente pase al siguiente *intent* y repita el bucle. En la Figura 3.8, se pueden ver como están definidas las frases de entrenamiento dentro de estas intenciones del bucle (en el ejemplo, se ven las del *intent* **Interview 1.2**):

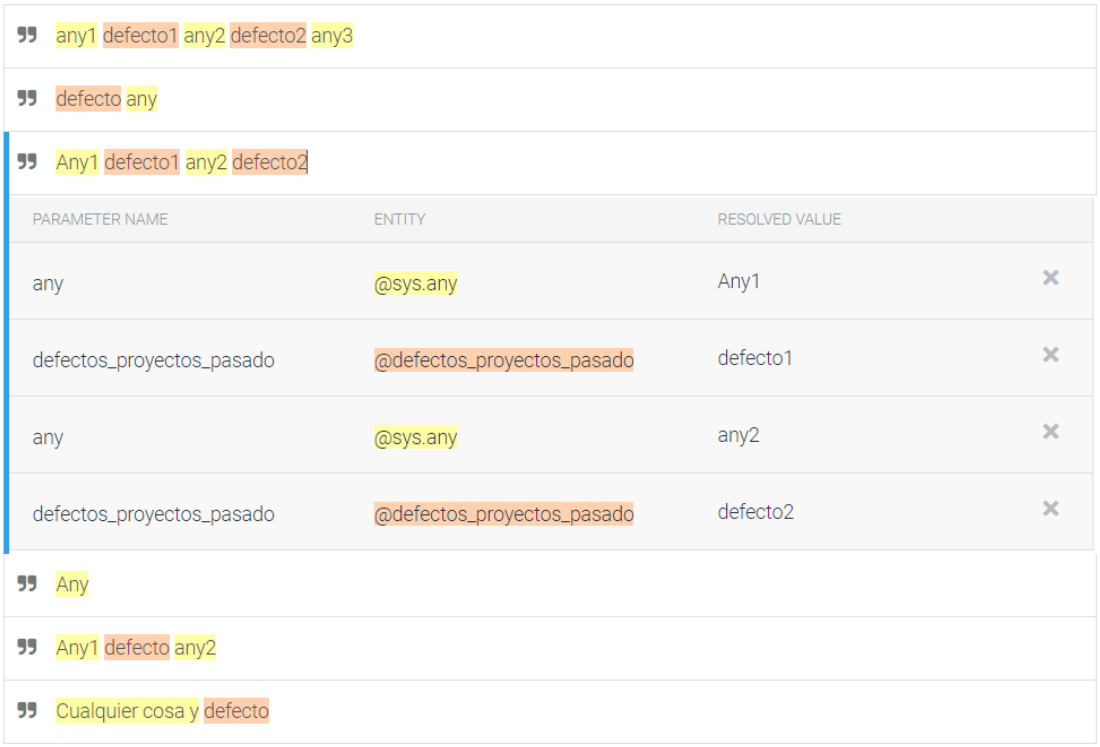


Figura 3.8: Frases de entrenamiento del *intent* Interview 1.2

Como se puede observar, hay ciertas diferencias respecto a anteriores frases de entrenamiento vistas en otras intenciones. La gran diferencia es la utilización de las variables, ya que es esta fase se realiza el reconocimiento de términos y expresiones clave para determinar el ratio de relevancia de la respuesta obtenida respecto a la pregunta anterior. Esta intención proviene de la pregunta encontrada en la Figura 3.6, donde al usuario se le pedía recopilar defectos que haya podido encontrar en anteriores proyectos. Puesto que las respuestas de los candidatos en una entrevista no tienen una estructura definida y son de estructura libre, las frases de entrenamiento no pueden ser frases muy concretas que puedas esperar que el candidato responda, ya que, en caso de definir frases de entrenamiento erroneas, el agente puede no reconocer la intención actual y volver al comienzo de la conversación. Es por ello que para la definición de las frases de entrenamiento de la fase de la entrevista utilizan la entidad @sys.any (encontrada en la figura anterior con un color amarillo). Esta entidad representa un comodín para la entrada de información, cualquier comentario realizado entrará en la

variable any, y como el flujo del diálogo está actualmente controlado por el agente, este sabe que cualquier comentario realizado irá a la siguiente intención del contexto de la entrevista. Entonces, como Dialogflow trata la respuesta como un texto del que no conoce el contenido, es necesario que, para encontrar estos términos clave, se definan entidades (como se verá en el siguiente apartado) que representen a estos términos, y que puedan estar en cualquier posición de la entrada (de ahí las múltiples frases de entrenamiento que solo mueven las variables de lado a lado). En este caso, la entidad definida es @defectos_proyectos_pasado, cuyo diccionario de términos incluyen una complicación de términos esperados para esta pregunta. Estas variables son de tipo array, por lo que se pueden encontrar varias en una misma entrada de texto, y se puede acceder a ellas fácilmente. Aunque solo se han definido algunas de las posibilidades del orden en el que se pueden encontrar los elementos dentro de un patrón de entrada, la capacidad de Dialogflow de adaptar sus modelos automáticamente permite que pueda encontrar cualquier combinación posible en los patrones de entrada sin necesidad de definirlos explicitamente en las frases de entrenamiento.

En la Figura 3.9, se muestran las respuestas generadas por un *intent* integrado en la fase de la entrevista, de la misma intención Interview 1.2 que la Figura 3.8:

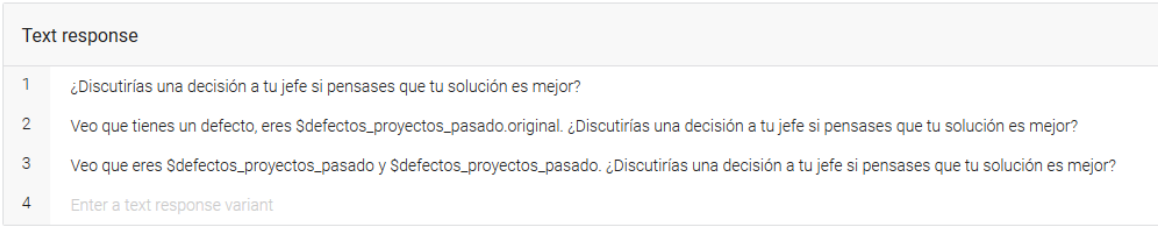


Figura 3.9: Respuestas posibles del *intent* Interview 1.2

Las respuestas de este *intent* plantean al usuario las nuevas preguntas que debe responder dentro de la simulación. En este caso, aunque la pregunta planteada es siempre la misma, se ofrece la posibilidad de utilizar las variables anteriormente declaradas con las palabras clave encontradas al recibir la respuesta del usuario, para enseñar al usuario que el falso entrevistador le escucha y sabe recoger los conceptos que va utilizando. El bucle de ejecución, tal y como fue definido anteriormente, termina cuando el agente de Dialogflow llega a la última pregunta de la entrevista (es decir, al último *intent* definido en la línea de entrevista elegida. El modelado del agente permite definir qué intención es la última de la conversación, y si es marcada, la comunicación con la aplicación utiliza un *flag* que indica el fin de la conversación. Cuando esto ocurre, el agente se apaga, y es necesario reiniciar la conexión con el agente para volver a conversar con él.

### 3.3.3. Entidades (*entities*)

Se ha podido ver como las intenciones hacen uso de las entidades para definir conceptos encontrados en la información de entrada. Como se pudo ver en el apartado anterior, Dialogflow contiene un alto número de entidades previamente definidas que

pueden ser utilizadas libremente por los desarrolladores pero, en caso de necesitar la utilización de términos personalizados, los desarrolladores del modelo pueden crear sus propias entidades manualmente. Para ello, Dialogflow permite utilizar su propia interfaz para definir estos diccionarios de palabras clave. La interfaz puede ser vista en la Figura 3.10:

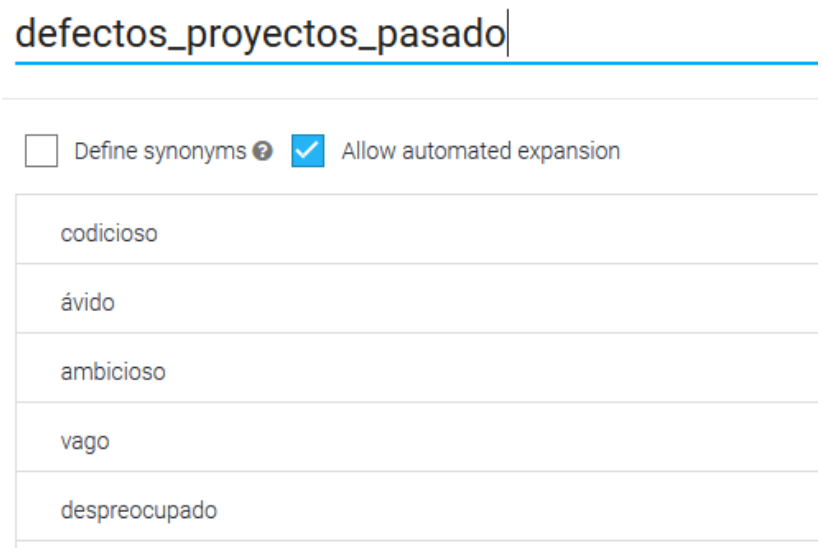


Figura 3.10: Interfaz de creación de entidades en Dialogflow

En el ejemplo encontrado en la figura anterior, se puede ver la definición de la misma entidad usada de ejemplo en el apartado anterior. Esta *entity* recoge un conjunto de términos que entrarían dentro de dicha entidad. En este caso, esta entidad se refiere a las palabras esperadas por un entrevistador cuando un usuario es preguntado por sus defectos en proyectos en sus trabajos anteriores, y en este caso son adjetivos que el usuario utilizaría para definirse a si mismo. La aparición de alguno de estos adjetivos dentro de la respuesta del usuario indica una relevancia sobre la pregunta realizada, lo cual es positivo para el análisis realizado aquí. Los servicios de análisis de sentimientos serían los encargados de mirar si el tono utilizado es el correcto, pero este análisis estudiaría el acercamiento de esta respuesta a la pregunta realizada, de la misma manera que un entrevistador real buscaría que las respuestas recibidas fueran útiles y relevantes para la conversación y no aleatorias.

Es necesario crear una entidad nueva por cada pregunta realizada por parte del agente al usuario, para poder concretar cuales son los términos esperados por el entrevistador virtual, por lo que la lista de términos puede acabar siendo muy grande. Sin embargo, como se puede observar en la Figura 3.10, hay una opción en la definición de entidades llamada *Allow automated expansion* o, en castellano, permitir expansión automática. Esta opción lo que permite es que los procesos de aprendizaje automático utilizados en el entrenamiento de las intenciones se aplique a las entidades también, y que, utilizando conexiones a Google, pueda reconocer sinónimos automáticamente que puedan ser añadidos a la lista, y así evitar que el desarrollador tenga que declarar todos los términos esperados manualmente. En el caso de este proyecto, donde un alto

número de entrevistas puede resultar en un alto número de entidades, ayuda a que el desarrollo de las entidades sea más sencillo.

### 3.3.4. Integración

La integración con el servicio de Dialogflow es algo distinta a la encontrada en los servicios de análisis de sentimientos. Mientras que estos permitían el uso de conexiones HTTP para transmitir información, Dialogflow requiere del uso de SDKs dedicados para poder trabajar con la API del agente. Para interactuar con esta API, además, es necesaria una clave para acceder a la API, que será utilizada por la aplicación que integre el SDK. En el caso de este proyecto, la aplicación estará en Android, y Dialogflow ofrece un SDK para Android de código abierto disponible para cualquier desarrollador que desee utilizarla.

Sin embargo, este SDK lleva tiempo sin ser actualizado y, aunque hay una versión más moderna para el lenguaje Java (el utilizado por la aplicación), requiere de configuración extra de la cuenta de Google Cloud donde el proyecto de Dialogflow reside que puede llevar a gastos extras. Es por ello que, para este proyecto, se utiliza el SDK antiguo, que todavía mantiene la posibilidad de conectarse correctamente con Dialogflow. La única desventaja de este SDK respecto a la versión más nueva es que solo funciona con la versión 1 de la API, mientras que Dialogflow ha comenzado a utilizar la versión 2 en los nuevos proyectos creados. Esta nueva versión 2 de la API de Dialogflow incluye cambios, principalmente, en la utilización del *fulfillment* (descrito en el Estado del arte, la conexión con servicios externos al agente para recuperar información extra) y en la definición del objeto JSON de respuesta. Ya que el SDK viejo es sencillo de implementar, y estas mejoras de la API no influyen al agente del proyecto, el agente está definido para utilizar la API V1.

Aunque todo el tratamiento del objeto JSON será realizado mediante funciones integradas en el SDK, conviene conocer los elementos más importantes de este objeto con los que, posteriormente, se trabajará en la aplicación. En la Tabla 3.7, se detallan los elementos más importantes de dicho objeto JSON para este proyecto:

| Clave | Descripción |
|---|---|
| parameters | Array de entidades encontradas en el texto analizado. Aquí la aplicación puede encontrar si el agente ha encontrado relaciones entre la pregunta y la respuesta, para realizar luego el análisis. |
| speech | La respuesta del agente, en un formato de texto plano. |
| intentName | Nombre de la intención reconocida, que puede ser útil para conocer, por ejemplo, el número de la pregunta realizada. |
| score | Confianza (representada con un valor del 0 al 1) que el agente tiene con la intención encontrada. |

Tabla 3.7: Estructura del JSON de respuesta de Dialogflow (API V1) [86]

## 3.4. Aplicación Android

El último gran módulo de este proyecto es el hilo conductor que junta todas las piezas de este proyecto, y el método principal de interacción que los usuarios tendrán para utilizar el sistema. Es la aplicación móvil, desarrollada para funcionar en la plataforma Android, y que su principal función es la de ofrecer un nexo de unión entre el usuario y los servicios de análisis de sentimientos, junto con el agente conversacional, para conformar el sistema de simulación de entrevistas de trabajo deseado.

La decisión de utilizar Android como plataforma de esta aplicación reside en varios factores. La primera y más importante es que, tal y como se explicó en el Estado del arte, Android es la plataforma móvil más utilizada en el mundo. La distribución de dispositivos con Android es mucho mayor que la encontrada en iOS o Windows Phone, y, por lo tanto, el marco de usuarios que potencialmente podrían utilizar la aplicación es más amplio. Por otro lado, estos dispositivos móviles ofrecen todas las características de hardware necesarias para conseguir realizar los objetivos descritos anteriormente y poder crear una aplicación multimodal, ya que son dispositivos que, en la actualidad, todos utilizan pantallas táctiles, micrófonos, cámaras y altavoces para poder interactuar con los usuarios, y así ofrecer a los desarrolladores la posibilidad de utilizar todos ellos para crear sus aplicaciones. Por último, destacar que, tal y como se podrá ver en el siguiente apartado, Android permite a los desarrolladores utilizar el lenguaje de programación Java para generar la lógica de las aplicaciones, siendo Java uno de los lenguajes de programación más utilizados mundialmente, por lo que la documentación es abundante (lo que ayuda mucho a los desarrolladores para crear sus aplicaciones). En los siguientes apartados, se definirán las herramientas utilizadas para el desarrollo de esta aplicación, además de una explicación del desarrollo de cada uno de los submódulos encontrados en la aplicación.

### 3.4.1. Herramientas utilizadas

El primer paso para poder desarrollar esta aplicación es tener acceso al SDK de Android, la interfaz de comunicación entre el sistema operativo y la lógica interna creada por el desarrollador de la aplicación. El SDK de Android ofrece la posibilidad de utilizar librerías y procesos de depuración para crear estas aplicaciones, y está disponible tanto para Windows 7 en adelante como para Linux y MacOS 10.6 en adelante. Además del SDK, Android ofrece un entorno de desarrollo (IDE) exclusivo para Android, llamado Android Studio. En la Figura 3.11, se puede ver la interfaz básica de Android Studio:

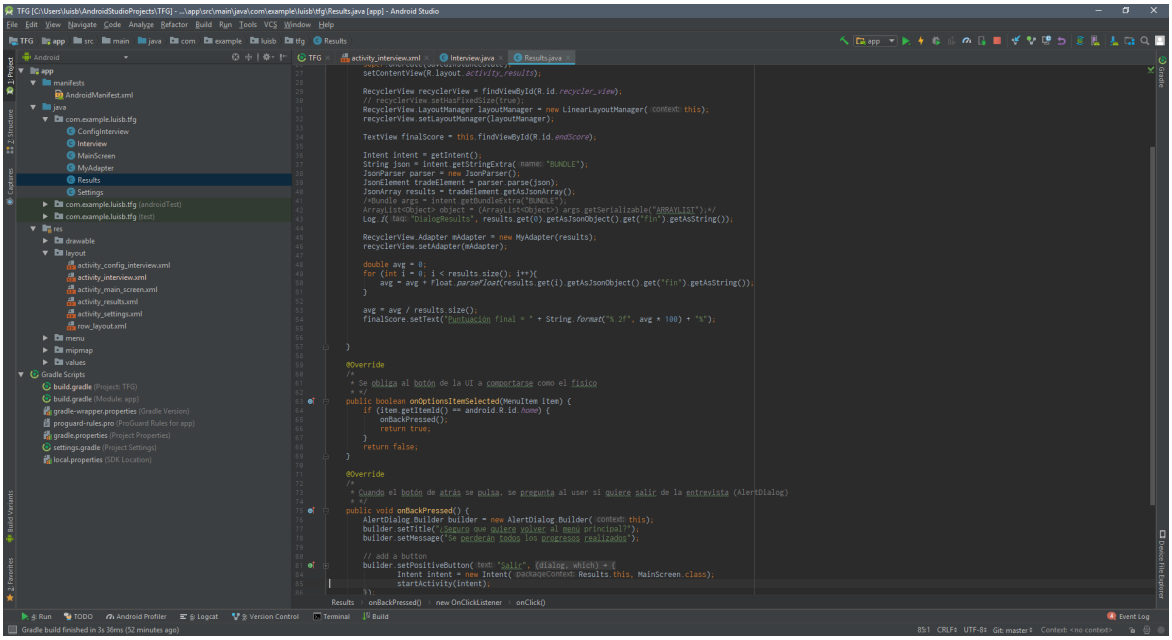

Figura 3.11: Interfaz básica de Android Studio

Android Studio es un entorno de desarrollo basado en otro IDE llamado IntelliJ IDEA, desarrollado por la empresa JetBrains. Android Studio es el IDE oficial para el desarrollo de aplicaciones en Android, y todas sus funcionalidades se basan en la utilización de las características del SDK de Android para asistir a los desarrolladores. El programa está diseñado para utilizar la sintaxis necesaria para interactuar con el SDK, y asiste al desarrollador en la utilización de esa sintaxis en múltiples maneras, como la definición de métodos básicos, la utilización de funciones del sistema operativo, o la utilización de librerías específicas de Android, sin que el desarrollador deba conocer estos pasos anteriormente, por lo que es una herramienta muy útil para desarrolladores que nunca han trabajado con Android. Otras herramientas incluidas en Android Studio incluyen una herramienta de depuración (que permite congelar la ejecución del código en sitios determinador por el desarrollador, y poder ver el uso de la memoria, la batería, las conexiones a Internet y el procesador para poder hacer un perfil del rendimiento de la aplicación en dichos sitios) o un emulador de Android (*Android Virtual Device*) (que permite a los desarrolladores ejecutar una máquina virtual cuyo sistema operativo es un dispositivo Android, que puede ser personalizado, tanto en tamaño de pantalla, como

en carácterísticas de hardware). Este emulador permite hacer uso, virtualmente, de todas las carácterísticas de Android: se puede utilizar cualquier aplicación disponible en la Play Store, y se pueden simular elementos como el reconocimiento de huellas, la cámara o el acelerómetro, por lo que los desarrolladores no necesitan utilizar un dispositivo real para desarrollar en Android. Además, este emulador permite generar un dispositivo virtual con cualquier versión de Android, incluyendo versiones preliminares de futuros lanzamientos, por lo que la evaluación del rendimiento de la aplicación en distintas versiones de Android es posible sin necesidad de tener acceso a dispositivos con dichas versiones. Para el desarrollo de esta aplicación, se ha utilizado Android Studio, realizando pruebas en dispositivos virtuales emulados por el IDE y también en dispositivos físicos, ambos utilizando la versión 7.0 (también llamada Nougat), que era la última versión estable al comienzo del desarrollo del proyecto.

Android Studio permite a los desarrolladores varias opciones a la hora de crear una nueva aplicación. Para empezar, al crear una aplicación de Android, los desarrolladores deben determinar cual es el nivel de API (o versión del SDK) objetivo. Este nivel de API, como se determinó en el Estado del arte, es equivalente a la versión de Android mínima para poder ejecutar la aplicación, y, gracias a la fragmentación de versiones de Android en los dispositivos, una versión alta de la API ofrece más funcionalidades, pero previene que un alto porcentaje de los dispositivos que no han conseguido estas versiones más modernas puedan utilizar la aplicación. Volviendo a la Figura 2.7, se puede ver como la gran mayoría de los usuarios (alrededor de un 85 %) de Android utilizan una versión equivalente o superior a la 5.0 (Lollipop). Es por ello que, para el desarrollo de esta aplicación, fue decidido que el nivel de API mínimo para poder ejecutar la aplicación fuese el nivel 21 (equivalente a la versión 5.0), permitiendo que, así, se puedan acceder a un alto número de características, incluyendo todas las necesarias para poder realizar una correcta simulación de entrevistas de trabajo.

Por otro lado, Android ofrece la posibilidad de utilizar dos lenguajes de programación distintos para escribir la lógica de los proyectos. El primero es Java, lenguaje que ha estado integrado en Android desde sus inicios, y que es uno de los lenguajes de programación más utilizados en el mundo. El segundo, sin embargo, es un lenguaje más moderno cuya integración en Android está disponible desde octubre de 2017, cuando Android Studio se actualizó para darle soporte. Este lenguaje se llama Kotlin, y, al igual que Java, se ejecuta dentro de la máquina virtual de Java, lo que permite que todas las versiones de Android puedan ejecutarlo. Kotlin es un lenguaje cuya sintaxis es más sencilla y fácil de aprender que Java, además de ofrecer más control en el manejo de excepciones y errores que con Java podrían bloquear la aplicación. Tanto Java como Kotlin pueden utilizar métodos del otro lenguaje, por lo que Android permite la utilización mixta de ambos dentro de una aplicación, y las librerías usadas por uno y otro lenguaje son intercambiables. Sin embargo, y como fue explicado anteriormente, para este proyecto se ha decidido utilizar únicamente Java para desarrollar la aplicación, puesto que, aunque Kotlin ofrece ciertas ventajas respecto a Java, Java es un lenguaje utilizado desde hace años por gran parte de la comunidad de desarrolladores, y encontrar documentación o ejemplos de integración con Android es más sencillo que con Kotlin, que lleva menos tiempo sirviendo como lenguaje oficial de Android.

Por último, destacar que para el control de versiones del código, se ha utilizado el sistema de control Git, cuya administración puede ser controlada directamente desde la propia interfaz de Android Studio. Utilizando Git, es posible utilizar un repositorio remoto para mantener versiones actualizadas del código en la nube, y así poder desarrollar en distintos dispositivos sin necesidad de mover el código de un ordenador a otro. El repositorio remoto utilizado ha sido Bitbucket, que permite tener repositorios privados ilimitados de manera gratuita, integrado en Android Studio y su sistema interno de control de versiones.

### 3.4.2. Menú principal

Para empezar, en la Figura 3.12 se puede ver el flujo de ejecución básico de la aplicación. Como se puede ver, es un flujo muy sencillo, pero que, como se podrá ver en el apartado del módulo de la entrevista, puede complicarse en la lógica programada detras de cada vista. Sin embargo, para el usuario, el flujo de ejecución es el siguiente:


Figura 3.12: Flujo básico de interacción de la aplicación Android

El primer submódulo encontrado en la aplicación es la pantalla principal. En esta pantalla, el usuario solo tiene la opción de comenzar una nueva entrevista. Siempre que se termina una entrevista y se han visto los resultados, se vuelve a esta pantalla. En la Figura 3.13, se puede ver como es esta interfaz:


Figura 3.13: Menú principal de la aplicación Android

Como se puede ver, esta interfaz solo contiene un único botón, que permite al usuario realizar una nueva entrevista. Este botón debe ser utilizado mediante la pantalla táctil del dispositivo. La lógica de la vista es también muy sencilla, ya que solo realiza dos funciones: la primera es cambiar de vista cuando el usuario pulsa el botón, y la segunda es comprobar que la aplicación tiene permisos para conectarse a Internet. Este paso es esencial para el correcto funcionamiento de la aplicación, ya que para simular una entrevista es necesario que la aplicación contacte tanto con los servicios de *sentiment analysis* como con el propio agente conversacional de Dialogflow, que es el que generará el flujo de la entrevista. En versiones relativamente nuevas de Android (a partir de la 6.0) esto requiere que el usuario acepte el permiso de utilizar Internet manualmente, mientras que en versiones anteriores a esta estos permisos eran concedidos por el sistema operativo al descargar la aplicación.

### 3.4.3. Entrevista

Esta vista es la vista principal de la aplicación. Aquí es donde el usuario interactuará principalmente con la aplicación, y donde toda la simulación de la entrevista de trabajo será realizada y mostrada. También es donde la gran mayoría de la lógica interna coge forma, puesto que esta pantalla contiene los métodos y funciones necesarias para interactuar con el usuario de manera multimodal, la conexión con los servicios externos de análisis de sentimientos y el agente conversacional, y la posterior evaluación realizada del rendimiento del usuario respecto a los resultados obtenidos por estos servicios. En la Figura 3.14, se muestra la pantalla que el dispositivo renderizará y que el usuario podrá utilizar para interactuar con el sistema de simulación de entrevistas:

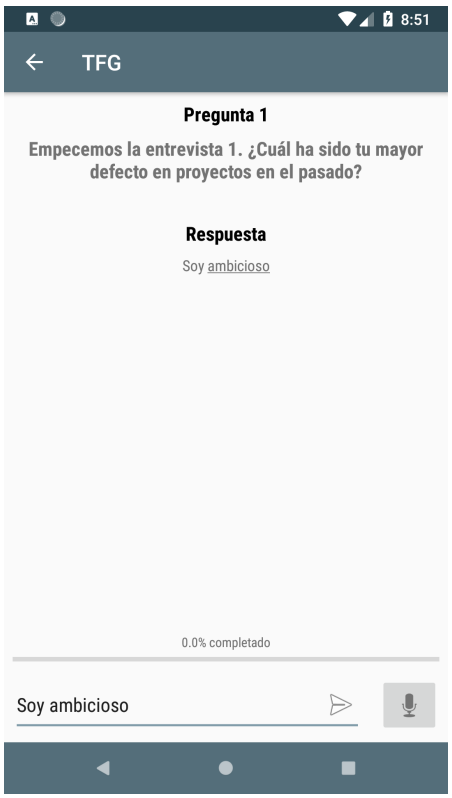


Figura 3.14: Pantalla de entrevistas de la aplicación Android

Como se puede ver, esta pantalla de interacción es, visualmente, muy sencilla y directa para el usuario. En la pantalla se ofrecen dos elementos principales: la pregunta realizada por el agente conversacional de Dialogflow (que se muestra como texto, pero también es reproducida, gracias a la utilización de un servicio *Text-To-Speech*, por la salida de audio del dispositivo, para que así el usuario puede reconocer al sistema como una voz humana que acerque su percepción de la simulación hacia un entorno real), y la respuesta del usuario. Esta respuesta se sitúa en la pantalla para que el usuario, en caso de utilizar técnicas de reconocimiento automático de habla para generar la respuesta, pueda ver el resultado de dicho análisis antes de enviarlo de vuelta al agente conversacional. Para la introducción de información, hay dos posibilidades, y el usuario puede decidir cual de ellas utilizar, de manera multimodal: o bien utiliza un campo de escritura libre, donde podrá utilizar un teclado para introducir la respuesta que quiere enviar al agente, o bien utiliza el micrófono del dispositivo para dictar la respuesta, y que el sistema de reconocimiento automático del habla de Android transforme esta respuesta a texto, que pueda ser enviado a los servicios externos. Además, se permite al usuario ver cual es el progreso de la entrevista, con un porcentaje de completitud de la entrevista en general, para que así puedan tener referencia de cuantas preguntas quedan para terminar la entrevista en un momento dado. Por último, destacar que el usuario siempre tiene la opción de salir de la entrevista, presionando el botón de retroceso integrado en la propia interfaz, o el equivalente integrado en todos los dispositivos Android, tal y como se puede ver en la Figura 3.15:

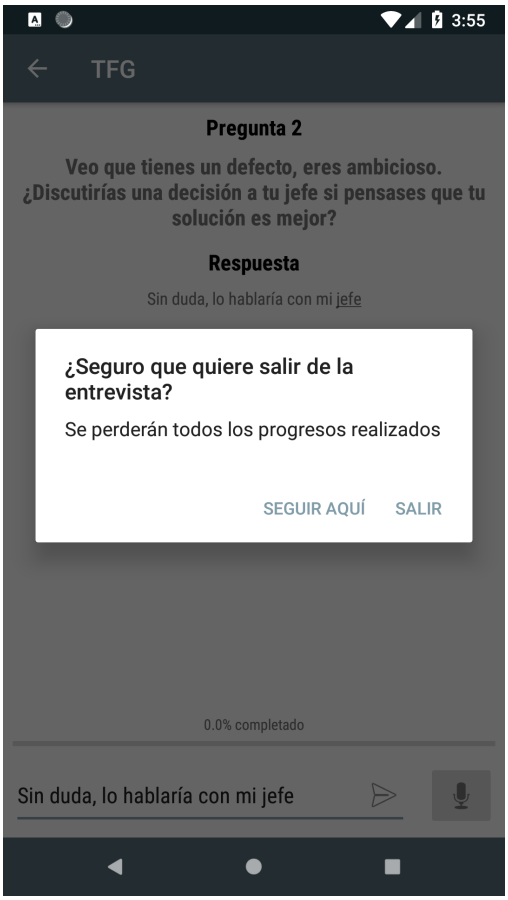


Figura 3.15: Salida de entrevistas de la aplicación Android

En la entrevista, los usuarios se encuentran en un bucle de ejecución, que para ellos, es equivalente a recibir una pregunta nueva, y responderla, y repetir estos dos pasos hasta que la entrevista termine. Sin embargo, internamente, este bucle implica un desarrollo lógico mayor, dado que la aplicación debe realizar varias tareas de manera transparente para el usuario para que este pueda realizar la tarea correctamente. En la Figura 3.16, se presenta un esquema de los pasos realizados internamente en esta vista para hacer que las entrevistas funcionen correctamente. Esta figura también podría definirse como una expansión del rectángulo central encontrado en la Figura 3.12, visto con más detalle.

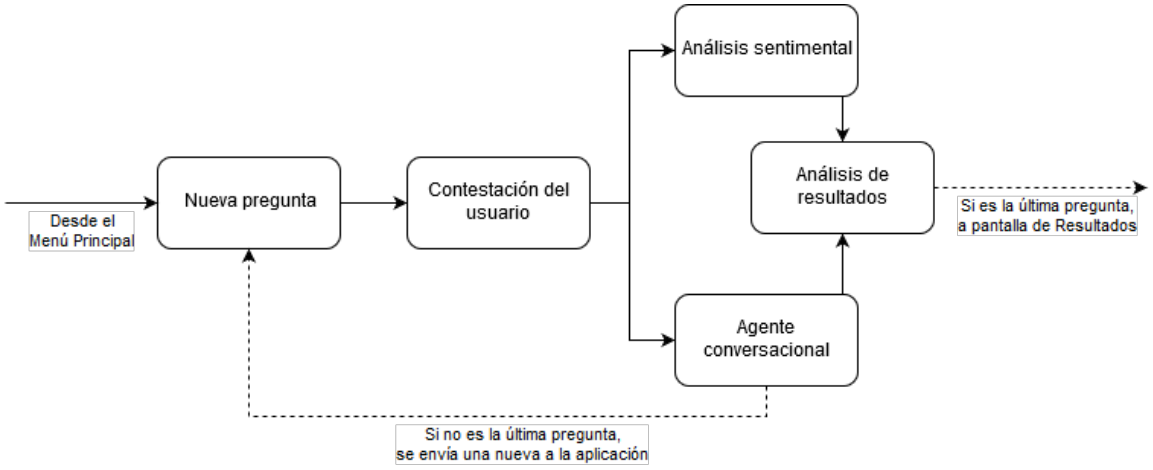


Figura 3.16: Lógica interna de la pantalla de Entrevista de la aplicación Android

Como se puede observar, la lógica interna de esta pantalla realiza múltiples acciones necesarias para realizar la simulación de la entrevista. Las acciones son las siguientes:

- **Nueva pregunta**. La aplicación recoge la nueva pregunta del agente conversacional (siempre y cuando no sea la última), y la transmite al usuario de dos maneras. La primera es que, utilizando el mismo texto que encuentra en la respuesta del agente, escribe por pantalla esta nueva pregunta, para que el usuario pueda verla visualmente y leerla. La segunda manera es, utilizando los módulos integrados en Android de síntesis de voz (*Text-To-Speech*), reproducir mediante la salida de audio del dispositivo una representación hablada de la pregunta generada por el agente, para que el usuario pueda escuchar la pregunta de la misma manera que lo haría en una entrevista real ante las preguntas propuestas por un entrevistador. Para realizar esta síntesis de voz, se ha utilizado la librería interna de Android llamada, adecuadamente, *TextToSpeech*, que se inicia al entrar a la vista y que puede sintetizar a voz cualquier texto que sea introducido en su función *speak*.

- **Respuesta del usuario**. El usuario, una vez leida o escuchada la pregunta realizada, puede responder a esta mediante dos métodos, similares a los encontrados en la recepción de la pregunta. Por un lado, el usuario puede utilizar un teclado (virtual o físico) para escribir, en un campo de texto libre, la respuesta que quiere enviar al agente conversacional. Por otro lado, el usuario también tiene la opción

de utilizar los servicios de reconocimiento automático del habla (*Speech-To-Text*) para generar su respuesta, simulando de una manera más fiel la interacción entre un candidato y un entrevistador en una entrevista real. Para conseguir este reconocimiento del habla, se ha integrado en la aplicación la librería interna de Android llamada *SpeechRecognizer*. Esta librería permite a la aplicación entrar en un estado especial, dirigido por el propio sistema operativo, que utiliza el hardware de entrada de sonido del dispositivo para realizar un reconocimiento del habla en directo, comunicándose al mismo tiempo con servidores externos de reconocimiento del habla de Google para obtener una mayor precisión en este proceso. Es por ello que para utilizar este sistema, el usuario debe permitir a la aplicación utilizar dos recursos del sistema: conexión a Internet (ya permitida al comenzar la entrevista) y acceso a los micrófonos del dispositivo. Ese estado particular de la aplicación superpone un mensaje del sistema operativo que informa al usuario del funcionamiento de la grabación, tal y como se puede ver en la Figura 3.17.


Figura 3.17: Utilización del servicio *SpeechRecognizer* en la aplicación Android

- **Conexión con servicios externos**. Una vez recibida la respuesta del usuario, la aplicación envía esta respuestas a los servicios externos que están integrados en el sistema. Estos servicios son los múltiples servicios de análisis de sentimientos vistos anteriormente y el propio agente conversacional que lleva el flujo de la conversación. Para la conexión con cada uno de ellos, ha sido necesario crear varias funciones distintas, que funcionan de manera paralela, y que contactan con estos servicios de distinta manera. Todas estas funciones se realizan de manera

asincrónica, utilizando las capacidades de Android de programación paralela y utilización de múltiples hilos de ejecución para realizar tareas (lo que, en las librerías de Android, se llama funciones *async*).

- **Analisis de sentimientos**. La conexión con cada uno de los servicios de análisis de sentimientos es distinta a la encontrada en los otros dos. Cada servicio, dadas las diferencias en las estructuras de los JSON de la petición y los SDKs utilizados, requiere de una configuración interna en la aplicación distinta, para poder enviarle a todas la respuesta del usuario en formato texto.

  MeaningCloud, al no tener un SDK disponible para Android, requiere de la utilización de una conexión HTTP configurada manualmente. Para ello, se utiliza la librería de peticiones HTTP integrada en Android (llamada *Volley*), que permite configurar una conexión HTTP hacia la dirección de la API especificada en la documentación de MeaningCloud, y definiendo el JSON de la petición también manualmente, según la estructura vista en la Tabla 3.1. Utilizando este método, la respuesta se recibe en la aplicación como un objeto JSON que puede ser manejado con la librería de Google *Gson*, que permite utilizar los JSON como clases y objetos.

  Por otro lado, Watson NLU permite la utilización de su SDK de Android, que fue importando previamente en el manifiesto de compilación de la aplicación como requisito para que la aplicación funcione. Este SDK tiene funciones integradas para generar el JSON de petición necesario y la conexión con la API de manera automática, por lo que es más sencillo de utilizar que una conexión HTTP manual como con MeaningCloud. La respuesta se recibe como un objeto personalizado del SDK llamado AnalysisResults, que contiene un alto número de funciones para poder leer cada uno de los parámetros de estos resultados fácilmente.

  Por último, Aylien utiliza un sistema similar al de MeaningCloud, ya que tampoco permite utilizar un SDK dedicado para contactar con la API de análisis de sentimientos, y es necesario hacer uso, de nuevo, de la librería *Volley* para realizar una petición HTTP manual a la API. El JSON de petición también debe ser creado manualmente, y en este caso la clave de contacto con la API debe estar asignada en la cabecera de la petición, en vez de dentro del cuerpo del JSON. La respuesta, al igual que MeaningCloud, viene en un objeto JSON que puede ser manejado utilizando la librería *Gson*.
- **Dialogflow**. Paralelamente a la conexión con los servicios de análisis de sentimientos, la aplicación contacta con la API del agente conversacional de Dialogflow para entregarle la respuesta del usuario. Esta conexión también se realiza en un hilo de ejecución específico para ello, para así no interrumpir el funcionamiento del resto de la aplicación. Para utilizar la API de Dialogflow, es necesario utilizar el SDK descrito en apartados anteriores, que contiene funciones tanto para configurar el agente (el idioma

esperado, la clave de la API necesaria, etc.), el envío de textos (el único parámetro esencial para enviar, en este caso, es la respuesta del usuario) y la recepción de resultados (que se mantiene dentro de un objeto especial llamado AIResponse, cuyos parámetros pueden ser accedidos mediante el uso de funciones especiales integradas en el SDK). En caso de encontrar un marcador de final de conversación en esta respuesta, la aplicación termina la conexión y pasa a la pantalla de Resultados.

- **Análisis de resultados**. En esta última fase, la aplicación recoge todos los resultados obtenidos de los servicios externos, y mediante el uso de un algoritmo saca conclusiones del rendimiento del usuario a la hora de responder a la última pregunta. El algoritmo devuelve un resultado del 0 al 1, donde el 0 representa el peor resultado posible, y el 1 el mejor. Este algoritmo se basa en dos partes: la primera es una media ponderada de las polaridades y las confianzas recibidas por los servicios de análisis de sentimientos. Para obtener esta media primero se normalizan las polaridades para tener un valor contenido entre el 0 y el 1 (en el caso de MeaningCloud, que solo ofrece 5 valores escrito en palabra, se van recorriendo estos valores en incrementos de 0.25 según el valor obtenido), y posteriormente estos valores son comparados con la respectiva confianza para conocer la polaridad final. Con esta polaridad final, se realiza una media ponderada de los tres valores. Mediante un proceso de prueba y error, se ajustan las ponderaciones para comprobar cual es el servicio más o menos acertado. En la versión final de la aplicación, esta ponderación era equivalente a 0.2 para MeaningCloud y 0.4 para Watson NLU y Aylien, ya que estos dos ofrecieron una precisión mayor respecto a los resultados esperados (esto se podrá ver con más detalle en el apartado de Evaluación). Por último, el análisis de Dialogflow se estudia mirando la cantidad de entidades personalizadas fue capaz de reconocer, y, en caso de haber reconocido palabras clave (que indican relación entre la respuesta y la pregunta), se suma al resultado 0.15. El resultado considera que cualquier valor inferior a 0.50 es una mala respuesta y superior o igual a 0.50 una buena respuesta. Este resultado se contiene dentro de un array de cada uno de los pasos de la entrevista entera, y en cada uno de estos pasos, se contiene la pregunta realizada, la respuesta generada, los valores de cada uno de los servicios de análisis por separado, la media de *sentiment analysis*, las palabras clave encontradas y el resultado final, para luego, en la siguiente pantalla, poder mostrar toda esta información por separado y así permitir al usuario conocer más detalles de su rendimiento.

Este bucle se ejecuta sin parar hasta llegar al final de la entrevista, que se identifica por una marca de fin de conversación enviada por el agente de Dialogflow. Una vez recibida a esta marca, la entrevista termina, y se pasa a la siguiente pantalla: la vista de Resultados. En el paso a esta pantalla se envia el array de resultados

### 3.4.4. Resultados

La última pantalla disponible es la pantalla de Resultados. En esta pantalla se recibe el array de resultados definido en la pantalla anterior y que, al terminar la

entrevista, fue enviado a esta pantalla. En la Figura 3.18, se puede observar la interfaz de esta parte de la aplicación, que enseña al usuario información relevante para conocer su rendimiento en la entrevista:

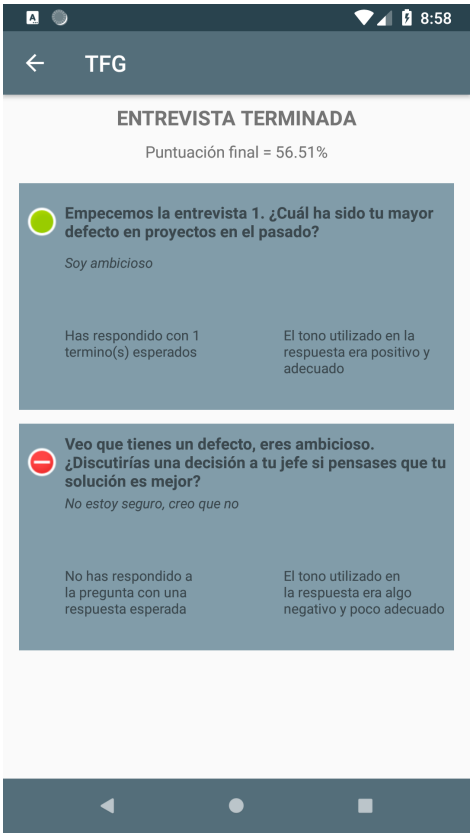


Figura 3.18: Pantalla de resultados de la aplicación Android

Esta pantalla contiene una multitud de elementos que representan el rendimiento del usuario al responder a las preguntas. Para empezar, se presenta una puntuación final (definida como un porcentaje del 0 al 100 %) que es calculada haciendo una media aritmética de los resultados finales de todas las respuestas realizadas. Con esta media, se puede averiguar el rendimiento global del usuario teniendo en cuenta todas las respuestas escogidas, sin ponderar unas respuestas unas por encima de otras. En este caso, un valor menor al 50 % se considera un mal rendimiento, mientras que un valor igual o superior al 50 % se considera un buen rendimiento.

Justo después de esta puntuación final se sitúa el desglose de respuestas del usuario. Aquí la aplicación muestra una lista de tarjetas, cada una correspondiendo a cada una de las respuestas realizadas por el usuario, que el usuario puede ir desplazando utilizando la pantalla táctil de su dispositivo. En ella se muestra varios elementos, empezando por la pregunta creada por el agente y la respuesta generada por el usuario, para que así este pueda saber cual es la respuesta que ha sido valorada. El principal elemento de esta tarjeta es un icono situado en la parte izquierda de la tarjeta. Este icono puede coger dos formas: un icono de color verde, que representa una respuesta correcta (es decir, un resultado de esa respuesta superior a 0.5), y otro de color rojo, que representa un resultado negativo (menor a 0.5) y por lo tanto una respuesta

incorrecta. Utilizando estos iconos, el usuario puede ver fácilmente si su respuesta ha sido adecuada o no. Sin embargo, además de esta indicación simple del rendimiento, cada tarjeta también ofrece dos comentarios extra, que detallan este rendimiento de cara al usuario. El primer comentario describe la relevancia de la respuesta respecto a la pregunta realizada (el análisis realizado por el agente de Dialogflow), explicando cuantos términos esperados ha encontrado. El segundo comentario explica al usuario el resultado obtenido del análisis de sentimientos realizado, comentando si la respuesta utilizada tenía un tono adecuado o demasiado negativo. Con estos comentarios, el usuario puede tener una idea de como mejorar sus respuestas en el futuro, sabiendo en que aspecto debe mejorar la siguiente vez que se enfrente a una pregunta similar.

Una vez el usuario ha terminado, la única opción disponible es la de volver al menú principal, donde podrá enfrentarse a otra entrevista de nuevo, y así volviendo a comenzar el bucle encontrado en la Figura 3.12.

## 3.5. Evaluación

En esta sección se explica la evaluación realizada a la aplicación, para comprobar que su funcionamiento es correcto, y los ajustes necesarios para conseguir la precisión necesaria para simular una entrevista correctamente.

### 3.5.1. Ajustes y evaluación del algoritmo

Dado que gran parte del funcionamiento del sistema depende de la precisión del algoritmo de análisis de resultados encontrado en el módulo de Entrevistas de la aplicación, es necesario hacer un estudio específico del funcionamiento de este algoritmo para conseguir una ponderación adecuada de cada uno de los resultados individuales de los servicios externos de *sentiment analysis* integrados en el sistema. Este algoritmo, como se puede ver en el apartado de desarrollo de la aplicación, es una media ponderada entre las polaridades y las confianzas de los resultados obtenidos en los análisis de sentimientos realizados por MeaningCloud, Watson NLU y Aylien, que acaba resultando en una polaridad final que representa el resultado final del usuario al responder a una pregunta.

Para ajustar estas ponderaciones, se ha utilizado la propia aplicación desarrollada para interactuar con el sistema, usando también los sistemas de depuración integrados en Android Studio para poder comprobar los resultados obtenidos por cada uno de los servicios externos por detrás de la aplicación en tiempo de ejecución. Para realizar las pruebas, la metodología seguida es la siguiente: dada una única pregunta ("¿Cuáles consideras que han sido tus mayores defectos en proyectos del pasado?") se han introducido varias respuestas, utilizando en cada una una intención y un tono distinto. Dado que este tono es conocido, es posible, viendo los resultados obtenidos por cada servicio, compararlo con los resultados de cada servicio para comprobar si han conseguido reconocer el tono o polaridad propuesta correctamente, y así poder ajustar la precisión del algoritmo. Los tonos planteados para esta prueba son los más comunes en una entrevista de trabajo: respuestas positivas, negativas y dubitativas y, como extra

dada la facilidad de los análisis de sentimientos es de no comprender la doble negación, un estudio extra utilizando esta doble negación. Los resultados se pueden ver en la Tabla 3.8 (los valores ya tienen calculada la media de la polaridad y la confianza de un conjunto de preguntas con el mismo tono):

| **Respuesta** | **MeaningCloud** | **WatsonNLU** | **Aylien** |
|---|---|---|---|
| Positiva | 0.789 | 0.814 | 0.792 |
| Positiva (doble negación) | 0.632 | 0.733 | 0.788 |
| Negativa | 0.484 | 0.259 | 0.311 |
| Negativa (doble negación) | 0.436 | 0.321 | 0.298 |
| Dubitativa | 0.412 | 0.520 | 0.502 |
| Dubitativa (doble negación) | 0.381 | 0.495 | 0.513 |

Tabla 3.8: Resultados de *sentiment analysis* obtenidos con varios tonos de respuestas

Como se puede ver en la tabla anterior, cada uno de los servicios de análisis de sentimientos consigue unos resultados distintos, pero hay uno que destaca en comparación a los otros dos. MeaningCloud es capaz de reconocer correctamente las frases más simples (sin ningún tipo de doble negación). Sin embargo, en cuanto la frase se complica y comienza a utilizar estructuras más complejas, como la doble negación estudiada, MeaningCloud deja de dar los resultados esperados y empieza a tener problemas para identificar correctamente la polaridad de la respuesta. Se puede observar, por ejemplo, en los dos tipos de respuestas positivas estudiadas, las frases simples las reconoce positivamente sin problema, pero introduciendo la doble negación la polaridad se reduce en un ratio mayor que los otros dos servicios (que, en comparación, mantienen una polaridad similar, especialmente Aylien que es particularmente acertada a la hora de reconocer dobles negaciones). Por otro lado, la duda está correctamente estudiada en los tres servicios, pero MeaningCloud, dado el modelo personalizado que fue introducido para dar polaridad negativa a las formaciones dubitativas, obtiene una polaridad más negativa que el resto de servicios, que se ve aumentada todavía más cuando se introducen estructuras lingüísticas más complejas., y no es consistente respecto al análisis realizado por Watson NLU y Aylien, que son más precisos en sus valoraciones. Watson NLU parece tener un rango de resultados algo mayor que los otros dos, aunque igualmente preciso, mientras que Aylien parece ser el más estable de los tres algoritmos.

Viendo estos resultados, fue decidido que, dada la inestabilidad de MeaningCloud en algunos análisis, su resultado no puede ponderar tanto como los resultados de Watson NLU y Aylien, que ofrecen más precisión y estabilidad que el primero. Por ello, como se puede ver en el apartado de desarrollo de la aplicación, el algoritmo pondera inferiormente a MeaningCloud (con un 0.2 de ponderación en la media realizada), mientras que a Watson NLU y Aylien, dados los resultados tan similares que ofrecen son ponderados de manera equivalente (con una ponderación de 0.4 cada uno).

En el siguiente apartado, se realizará un estudio del rendimiento de este algoritmo, junto con las pruebas necesarias para comprobar el funcionamiento del sistema global.

### 3.5.2. Pruebas

Para comprobar el correcto funcionamiento del sistema, se ha realizado una multitud de pruebas utilizando la aplicación de Android, que es el punto de entrada de los usuarios al sistema. Dada la subjetividad de los resultados del sistema de simulación de entrevistas, las pruebas deben ser realizadas conociendo un resultado esperado (al igual que en el estudio del algoritmo), ya que, potencialmente, las respuestas consideradas acertadas por un usuario pueden no son consideradas acertadas en este sistema, lo que provocaría que la evaluación del usuario fuera negativa.

A continuación se detallan las pruebas funcionales realizadas al sistema, siguiendo la estructura de la Tabla 3.9, que define un identificador para la prueba, el nombre del elemento probado, una descripción del elemento, los pasos a seguir y el resultado esperado:

| **Identificador** | **PF-XX** |
|---|---|
| **Nombre** | |
| **Descripción** | |
| **Pasos** | |
| **Resultado esperado** | |

Tabla 3.9: Tabla de ejemplo de pruebas funcionales

Las pruebas son las siguientes (definidas desde la Tabla 3.10 hasta la Tabla 3.22), y fueron realizadas tanto en el emulador de dispositivos virtuales integrado en el SDK de Android y Android Studio, como en un dispositivo Android físico, ambos utilizando la versión 7.1 de Android:

| **Identificador** | **PF-01** |
|---|---|
| **Nombre** | Inicio de aplicación |
| **Descripción** | La aplicación debe ser capaz de iniciarse en un dispositivo Android. |
| **Pasos** | Utilizando un dispositivo Android, se debe pulsar el icono de la aplicación instalada. |
| **Resultado esperado** | La aplicación se inicia y sale el menú principal. |

Tabla 3.10: PF-01 Inicio de aplicación

| **Identificador** | **PF-02** |
|---|---|
| **Nombre** | Aceptación de permisos de Internet |
| **Descripción** | En caso de que sea la primera vez que el usuario ejecuta la aplicación en un dispositivo con Android 6.0 o superior, es necesario que el usuario acepte la petición de la aplicación de utilizar Internet. |
| **Pasos** | Desde el menú principal, se pulsa en la tarjeta "Realiza una entrevista". El sistema muestra un cuadro de diálogo pidiendo al usuario permisos para utilizar la red del dispositivo, la cual debe ser aceptada. |
| **Resultado esperado** | La entrevista comienza. |

Tabla 3.11: PF-02 Aceptación de permisos de Internet

| **Identificador** | **PF-03** |
|---|---|
| **Nombre** | Comienzo de entrevista |
| **Descripción** | El usuario, desde el menú principal, debe ser capaz de acceder a una nueva entrevista. |
| **Pasos** | Desde el menú principal, se pulsa en la tarjeta "Realiza una entrevista" (y, potencialmente, se deben aceptar los permisos de Internet, visto en la prueba PF-02). |
| **Resultado esperado** | La aplicación muestra la vista Entrevista. |

Tabla 3.12: PF-03 Comienzo de entrevista

| **Identificador** | **PF-04** |
|---|---|
| **Nombre** | Muestra de nueva pregunta por pantalla |
| **Descripción** | El usuario debe ser capaz de, una vez comenzada la entrevista o responder a una pregunta, ver por pantalla una representación textual de una nueva pregunta realizada por el sistema. |
| **Pasos** | Comenzar una entrevista o responder a una pregunta utilizando los medios disponibles. |
| **Resultado esperado** | Una nueva pregunta se muestra por pantalla. |

Tabla 3.13: PF-04 Muestra de nueva pregunta por pantalla

| **Identificador** | **PF-05** |
|---|---|
| **Nombre** | Reproducción de nueva pregunta por salida de sonido |
| **Descripción** | Utilizando una voz sintetizada, la aplicación debe ser capaz de utilizar la salida de sonido del dispositivo para reproducir una nueva pregunta generada por el sistema. |
| **Pasos** | Comenzar la entrevista o responder a una pregunta. |
| **Resultado esperado** | La pregunta es comunicada al usuario utilizando la salida de audio del dispositivo, mediante el uso de síntesis de voz. |

Tabla 3.14: PF-05 Reproducción de nueva pregunta por salida de sonido

| **Identificador** | **PF-06** |
|---|---|
| **Nombre** | Introducción de respuestas por pantalla |
| **Descripción** | El usuario debe ser capaz de introducir, mediante teclado, la respuesta que quiere enviar al sistema. |
| **Pasos** | En una entrevista, pulsar en el campo de escritura libre encontrado en la pantalla, utilizar un teclado virtual o físico para escribir la respuesta deseada y pulsar en el botón de envío situado a la derecha del campo de escritura. |
| **Resultado esperado** | El sistema acepta la nueva respuesta y se muestra un cuadro de carga que es mostrado hasta que los análisis externos son realizados. En cuanto esta carga termina, se muestra la nueva pregunta al usuario. |

Tabla 3.15: PF-06 Introducción de respuestas por pantalla

| **Identificador** | **PF-07** |
|---|---|
| **Nombre** | Introducción de respuestas por voz |
| **Descripción** | El usuario debe ser capaz de dictar la respuesta deseada a la aplicación, y esta reconocerla correctamente. |
| **Pasos** | En una entrevista, pulsar el botón del micrófono encontrado en la interfaz, expresar la respuesta utilizando la voz al dispositivo y, al terminar, parar de hablar. |
| **Resultado esperado** | El sistema acepta la nueva respuesta y se muestra un cuadro de carga que es mostrado hasta que los análisis externos son realizados. En cuanto esta carga termina, se muestra la nueva pregunta al usuario. |

Tabla 3.16: PF-07 Introducción de respuestas por voz

| **Identificador** | **PF-08** |
|---|---|
| **Nombre** | Muestra de la respuesta por pantalla |
| **Descripción** | Antes de enviar la respuesta, el usuario debe ser capaz de ver por pantalla la respuesta que va a ser enviada. |
| **Pasos** | Escribir o dictar una respuesta. |
| **Resultado esperado** | La respuesta generada es mostrada en la interfaz. |

Tabla 3.17: PF-08 Muestra de la respuesta por pantalla

| **Identificador** | **PF-09** |
|---|---|
| **Nombre** | Terminar entrevista |
| **Descripción** | Una vez llegada al final de una entrevista, la aplicación debe ser capaz de terminar la conexión con los servicios externos y pasar a la pantalla de resultados. |
| **Pasos** | Responder a todas las preguntas de una entrevista. |
| **Resultado esperado** | La aplicación pasa a la vista de Resultados. |

Tabla 3.18: PF-09 Terminar entrevista

| **Identificador** | **PF-10** |
|---|---|
| **Nombre** | Volver al menú principal (Entrevista) |
| **Descripción** | Desde la vista de Entrevista, el usuario debe ser capaz de utilizar el botón físico del dispositivo o el botón integrado en la interfaz para poder volver al menú principal. |
| **Pasos** | Durante una entrevista, el usuario presiona el botón integrado en la interfaz para ir atrás o el propio botón físico del dispositivo, y acepta salir en el cuadro de alerta enseñado por pantalla. |
| **Resultado esperado** | La aplicación muestra el menú principal. |

Tabla 3.19: PF-10 Volver al menú principal (Entrevista)

| **Identificador** | **PF-11** |
|---|---|
| **Nombre** | Muestra de resultado final global por pantalla |
| **Descripción** | En la pantalla de Resultados, el usuario debe poder ver su resultado final global por pantalla. |
| **Pasos** | Terminar una entrevista y llegar a la pantalla de Resultados. |
| **Resultado esperado** | La aplicación muestra en su interfaz el resultado final global como un porcentaje. |

Tabla 3.20: PF-11 Muestra de resultado final global por pantalla

| **Identificador** | **PF-12** |
|---|---|
| **Nombre** | Muestra de resultados individuales por pantalla |
| **Descripción** | El usuario debe poder ver los resultados individuales de cada una de las respuestas realizadas para poder conocer con detalle su rendimiento en la entrevista. |
| **Pasos** | Terminar una entrevista y llegar a la pantalla de Resultados. |
| **Resultado esperado** | La interfaz muestra una tarjeta por cada respuesta realizada, mostrando la pregunta realizada, la respuesta del usuario, un indicador genérico del resultado de la respuesta, y un desglose de la relación entre la pregunta y la respuesta y el tono utilizado al formar la respuesta. |

Tabla 3.21: PF-12 Muestra de resultados individuales por pantalla

| **Identificador** | **PF-13** |
|---|---|
| **Nombre** | Volver al menú principal (Resultados) |
| **Descripción** | Desde la vista de Resultados, el usuario debe ser capaz de utilizar el botón físico del dispositivo o el botón integrado en la interfaz para poder volver al menú principal. |
| **Pasos** | En la pantalla de Resultados, el usuario presiona el botón integrado en la interfaz para ir atrás o el propio botón físico del dispositivo, y acepta salir en el cuadro de alerta enseñado por pantalla. |
| **Resultado esperado** | La aplicación muestra el menú principal. |

Tabla 3.22: PF-13 Volver al menú principal (Resultados)

### 3.5.3. Resultados

Una vez definidas las pruebas, en la Tabla 3.23 podemos observar los resultados obtenidos al realizarlas dentro del entorno de pruebas descrito anteriormente:

| **Identificador** | **Resultado** |
|---|---|
| PF-01 | Éxito |
| PF-02 | Éxito |
| PF-02 | Éxito |
| PF-04 | Éxito |
| PF-05 | Éxito |
| PF-06 | Éxito |
| PF-07 | Éxito |
| PF-08 | Éxito |
| PF-09 | Éxito |
| PF-10 | Éxito |
| PF-11 | Éxito |
| PF-12 | Éxito |
| PF-13 | Éxito |

Tabla 3.23: Resultados finales de las pruebas

Se puede observar como todas las pruebas funcionales han obtenido un resultado exitoso, por lo que se considera que la aplicación consigue realizar todos sus objetivos correctamente, siendo capaz de ofrecer a los usuarios la capacidad de trabajar e interactuar con el sistema de simulación de entrevistas tal y como se definió en los objetivos del proyecto.

Por otro lado, dado que el algoritmo de *sentiment analysis* del sistema es una pieza importante de la aplicación, y que ha requerido realizar un ajuste personalizado de

las ponderaciones para aumentar su precisión, se ha realizado una evaluación a los resultados de este algoritmo. Se han utilizado las mismas preguntas y banco de respuestas realizadas en el apartado anterior, para comprobar si la ponderación realizada entonces era correcta. En la Tabla 3.24, se desglosan estos resultados utilizando los mismos tonos de prueba que al ajustar el algoritmo:

| **Respuesta** | **Resultado** |
|---|---|
| Positiva | 0.799 |
| Positiva (doble negación) | 0.739 |
| Negativa | 0.353 |
| Negativa (doble negación) | 0.335 |
| Dubitativa | 0.495 |
| Dubitativa (doble negación) | 0.485 |

Tabla 3.24: Resultados del algoritmo ponderado con varios tonos de respuestas

Como se puede observar, los valores son más certeros con esta nueva ponderación (valores altos para comentarios positivos, bajos para los negativos, relativamente neutrales pero algo negativos para los dubitativos y una doble negación más consistente con las frases más simples), y la inestabilidad provocada por MeaningCloud ha sido reducida gracias a la utilización de una ponderación mayor con los otros dos servicios, que dan resultados más precisos. Con estos resultados, el sistema puede obtener una buena representación del análisis de sentimientos de las respuestas, lo cual lleva a resultados más acertados que enseñar al usuario y que conllevan un mejor sistema de retroalimentación para que el usuario pueda aprender a realizar entrevistas laborales.

# Capítulo 4

# Gestión del proyecto

En este apartado se definirá la planificación y gestión de todos los recursos disponibles para la realización del proyecto. En los siguientes apartados, se especificará cual fue la planificación temporal al comienzo del proyecto, el presupuesto utilizado para la realización de este y los marcos reguladores y socioeconómicos en los que se encuentra.

## 4.1. Planificación temporal

Para empezar, en este apartado se detallará la planificación temporal realizada al comienzo del proyecto. Esta planificación está relacionada a las fases principales del desarrollo de este proyecto, que son las siguientes:

- **Planificación del proyecto**. En esta fase, fue realizada toda la planificación e investigación de cada uno de los módulos que componen el proyecto y de las fases a realizar. Estos pasos son la definición del proyecto y el problema estudiado, un estudio de las funcionalidades de Android, investigación acerca de los sistemas de diálogo hablado, los análisis de sentimientos y los agentes conversacionales y, finalmente, un modelado de la solución a desarrollar una vez conseguidos todos estos conocimientos.
- **Desarrollo del sistema**. Una vez la planificación es terminada, se comienza a realizar el desarrollo de la solución propia, desarrollando toda aplicación Android y los modelos necesarios de los servicios externos, e integrando todos estos servicios externos a la aplicación para terminar de completar la solución.
- **Pruebas y evaluación**. Una vez terminado el sistema, es necesario probar todos sus componentes para conocer la efectividad de su funcionamiento.
- **Redacción de la memoria**. La escritura de este mismo documento.

Para definir los tiempos necesarios para cada fase del proyecto, se ha realizado un diagrama Gantt, que define visualmente el tiempo previsto que se va a dedicar para cada una de las fases del proyecto. Para realizar este diagrama, se ha utilizado el software GanttProject [87], que permite crear diagramas de Gantt sin problemas de licencia de ningún tipo. Cada fase, junto con cada una de las fechas estimadas para cada una de ellas, se puede ver en la Figura 4.1, junto con el tiempo (en días) que

se estima que cada fase y tarea necesiten para llevarse a cabo. Esta estimación en duración tiene en cuenta que, más o menos, se utiliza entre dos o cuatro horas al día para trabajar en este proyecto, y que no en todos los días del calendario se podrá avanzar en las tareas propuestas. La fecha de comienzo y de final corresponden a las fechas de primer contacto con el tutor para la asignación del trabajo y la fecha estimada de entrega de este documento, respectivamente.

| Nombre | Fecha de i... | Fech... | Dur... |
|---|---|---|---|
| Inicio del proyecto | 5/09/17 | 5/09/17 | 1 |
| Planificación | 6/09/17 | 13/02/18 | 115 |
| Definición del proyecto | 6/09/17 | 17/10/17 | 30 |
| Estudio de funcionalidades de Android | 18/10/17 | 7/11/17 | 15 |
| Investigación de sistemas de diálogo hablado | 8/11/17 | 28/11/17 | 15 |
| Definición de servicios de análisis sentimental | 29/11/17 | 12/12/17 | 10 |
| Búsqueda de agentes conversacionales | 13/12/17 | 26/12/17 | 10 |
| Modelado de la solución | 27/12/17 | 13/02/18 | 35 |
| Desarrollo | 14/02/18 | 10/05/18 | 62 |
| Creación del entorno de desarrollo | 14/02/18 | 15/02/18 | 2 |
| Desarrollo de los servicios externos | 16/02/18 | 19/04/18 | 45 |
| Desarrollo de la aplicación | 16/02/18 | 19/04/18 | 45 |
| Integración con sistemas externos | 20/04/18 | 10/05/18 | 15 |
| Pruebas y evaluación | 11/05/18 | 31/05/18 | 15 |
| Redacción de la memoria | 1/05/18 | 18/06/18 | 35 |

Figura 4.1: Estimación de tiempos de las fases del proyecto

En la Figura 4.2, está el propio diagrama, señalizando las fases principales del proyecto:

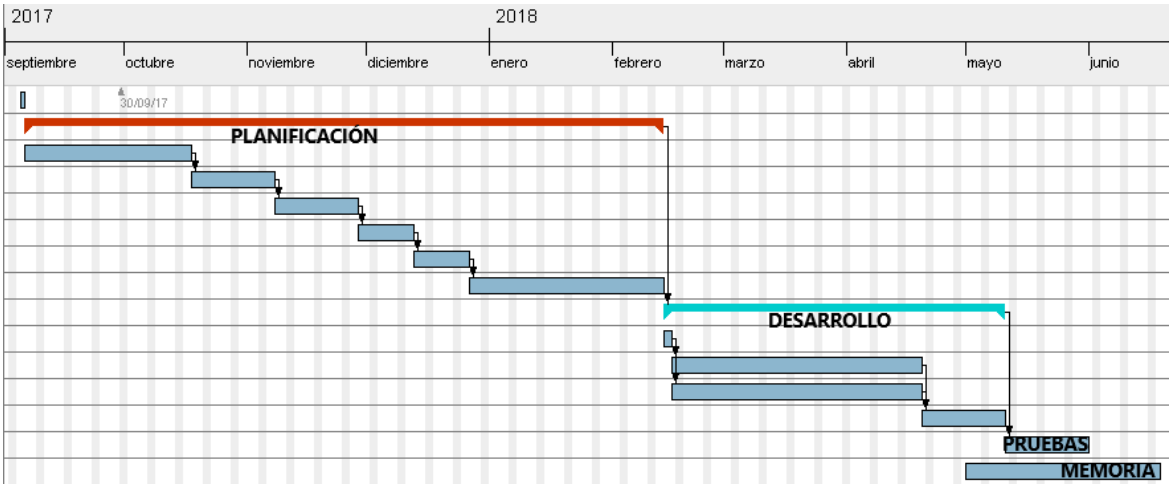


Figura 4.2: Diagrama Gantt de la planificación del proyecto

En la anterior figura, se puede ver como se conforma un diagrama de Gantt. En el diagrama, se recorre el tiempo de izquierda a derecha (en esta figura, comienza en septiembre en la izquierda, y acaba a finales de junio en la derecha), y cada una de

las tareas se representan por cada una de las filas determinadas en el diagrama (y que son equivalentes al orden encontrado en la Figura 4.1). Para determinar que una tarea entra dentro de una fase, la fase se convierte en una llave que recoge todas las tareas que encuentre debajo suyo. Por ejemplo, en este caso, la fase de planificación tiene una forma de llave y un color rojo, y todas las tareas que tiene debajo son las tareas que componen esta fase. Además, se especifica la dependencia de tares pasadas con una flecha (por ejemplo, en este diagrama, la fase de desarrollo no se puede realizar hasta terminar la fase de planificación, taly como marca la flecha que hay entre ambas fases).

Hay que destacar que una vez terminado el proyecto, esta planificación no ha sido cumplida completamente. Aunque algunas estimaciones han sido correctas, algunas tareas (como parte del desarrollo) han sido comenzadas paralelamente a otras tareas que, en teoría, debían ser requisitos para poder realizar tareas posteriores. Por ejemplo, aunque el desarrollo no debía comenzar hasta terminar la planificación, parte de este desarrollo fue realizado durante la investigación de, por ejemplo, el entorno Android, para ir avanzando mientras se aprendían conceptos. En conjunto, la estimación total del diagrama sí ha sido correcta, aunque con algunas tareas durando más o menos de lo esperado, pero equilibrándose con el tiempo de otras. En todo caso, siguiendo la planificación anterior, y teniendo en cuenta que las horas utilizadas cada día está entre dos y cuatro horas, más o menos, da el siguiente resultado (utilizando una media de 3 horas por día, y teniendo en cuenta que hay algunas fases paralelas, pero que implica que es necesario en esos casos realizar más tiempo para poder realizar las tareas encontradas):

| **Fase** | **Días estimados** | **Horas estimadas (3h/día)** |
|---|---|---|
| Inicio del proyecto | 1 día | 3 horas |
| Planificación | 115 días | 345 horas |
| Desarrollo | 62 días | 186 horas |
| Pruebas | 15 días | 45 horas |
| Redacción de la memoria | 35 días | 105 horas |
| **Total** | **228 días** | **684 horas** |

Tabla 4.1: Cálculo de tiempos estimados del proyecto

En resumen, tal y como se puede observar en la Tabla 4.1, se estima realizar un conjunto de 684 horas en la creación de este proyecto, pero, tal y como se ha declarado anteriormente, es posible que este número, en el desarrollo del proyecto final no sea exacto, por la posibilidad de no trabajar todos los días o algún día trabajar más horas de lo previsto.

## 4.2. Marco regulador

En esta sección, se definirá el marco legal en el que se encuentra este proyecto. Con esta definición, se obtendrá una visión de la legislación aplicada al proyecto, que

incluye conceptos como un análisis de la privacidad y la seguridad del sistema, además de un estudio de los términos de uso de todas las tecnologías utilizadas por el sistema.

El primer análisis realizado es un estudio de la legislación adherida a un proyecto de estas características. En una entrevista de trabajo, el entrevistador pregunta al candidato información relativa a su experiencia pasada tanto en educación como en otros entornos laborales, y, dada la intención de realizar una simulación realista de este proceso, el sistema de simulación de entrevistas de este proyecto plantea preguntas similares al usuario., por lo que las respuestas recibidas por el sistema, potencialmente, pueden ser privadas. Esto conlleva que la aplicación deba regirse por varios frentes legales, empezando por la ley de España. En España, la Constitución Española marca la protección de estos datos como un derecho constitucional de todos los ciudadanos del pais, en el Artículo 18.4 [88]:

> "La ley limitará el uso de la informática para garantizar el honor y la intimidad personal y familiar de los ciudadanos y el pleno ejercicio de sus derechos."

Con este artículo, la Constitución Española ratifica la necesidad de proteger la información personal de los ciudadanos en el ámbito informático, como es el de este proyecto. Esto es complementado con la Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal [89], una ley orgánica que busca proteger la intimidad y privacidad personal de los ciudadanos, aplicada también a un ámbito informático, prohibiendo el guardado y distribución de este tipo de información personal sin consentimiento explícito del usuario a las empresas situadas en España. Esta ley, sin embargo, durante el desarrollo del proyecto ha sido sustituida por una ley de ámbito europeo, promovida por la Unión Europea, llamada Reglamento General de Protección de Datos (RGPD en castellano, pero más conocida globalmente como GDPR) [90]. Fue aprobada en 2016, pero no entró en vigor hasta el 25 de mayo del 2018, dejando un margen de dos años a las empresas para adaptar su modelo de datos a esta nueva normativa. Este nuevo reglamento contiene todas las protecciones al consumidor encontradas en las leyes a las que sustituye, pero el elemento más importante es que ahora es aplicable a empresas externas a la Unión Europea que tratan con información de ciudadanos europeos, por lo que cualquier empresa introducida en Europa debe seguir esta ley de protección de datos. En este reglamento, los usuarios pueden explicitamente decidir cuales son los derechos de la empresa para utilizar esta información personal.

En el ámbito de este proyecto, la aplicación Android nunca guarda información personal de los usuarios permanentemente. En la aplicación, la información es guardada únicamente durante el tiempo requerido para que el usuario termine la entrevista actual, pero una vez terminada esta, la información es borrada, y ni el sistema ni los propios usuarios pueden volver a acceder a ella, lo que sigue el reglamento de protección de datos visto anteriormente. Sin embargo, es cierto que la aplicación envía esta información a servicios externos, que pueden tener otros términos de uso que puedan ser contrarios a este reglamento. En la Tabla 4.2, se especifican las protecciones de privacidad encontradas en cada uno de los servicios externos utilizados:

| **Servicio** | **Conformidad con GDPR** | **Notas** |
|---|---|---|
| Speech-To-Text de Android | Si | Utiliza los términos de privacidad de Google, que monitoriza toda la información recibida en sus APIs y la utilizará para mejorar sus servicios [91]. |
| Dialogflow | Si | Al igual que el Speech-To-Text anterior, utiliza los términos de privacidad de Google, para mejorar sus servicios en el futuro [91]. |
| MeaningCloud | Si | Guarda los textos analizados durante un tiempo limitado, pero no permite que sean accedidos excepto para comprobar el correcto funcionamiento del sistema [92]. |
| Watson NLU | Si | Utiliza los términos de servicio de IBM Cloud, que, al igual que MeaningCloud, no utiliza la información recibida excepto para realizar mantenimientos en el sistema [93]. |
| Aylien | Si | Aunque Aylien conserva los textos analizados, no es posible acceder a ellos y Aylien no los comunica a empresas externas [94]. |

Tabla 4.2: Comparación de los términos de privacidad de los servicios externos

En resumen, los términos de los servicios externos permiten a esta aplicación conectarse a ellos sin necesidad de preocuparse por la privacidad de los contenidos generados por los usuarios, ya que entran dentro del marco legal principalmente marcado por el GDPR, y que protege a los usuarios de la posibilidad de que su información personal sea utilizada erróneamente. Hay que destacar también que los usuarios, más allá de los contenidos de las respuestas que realicen, no introducen ningún otro tipo de información dentro de la aplicación, por lo que, además, la información enviada existe sin ningún tipo de contexto del usuario que utiliza la aplicación. Como extra, para utilizar los planes gratuitos de MeaningCloud y Aylien, es obligatorio, según sus términos de uso, mostrar en la aplicación el logo corporativo de cada uno de ellos, y un enlace a la página principal de ambos servicios. En el caso de la aplicación del proyecto, estos logos se encuentran en una pantalla no esencial para el funcionamiento de la aplicación, que contiene los créditos de la aplicación.

Por otro lado, además de los servicios externos, Android tiene unos términos concretos para los desarrolladores, que deben seguir unas guías para poder crear aplicaciones en la plataforma. La principal norma es la necesidad de ofrecer a todos los usuarios la misma funcionalidad, independientemente de su localización, y el derecho a utilizar la aplicación permanentemente en su dispositivo. Además, para poder distribuir la

aplicación en la Play Store, es necesario que la aplicación no incumpla ninguna ley de los mercados en los que esté disponible y que respete los términos de privacidad de Google (también utilizados por Dialogflow) para asegurar la privacidad y la seguridad de los usuarios de la aplicación. Actualmente, la aplicación, al asegurarse que toda la transmisión de información se adhiere a los términos propuestos por el GDPR, y no imponer ningún tipo de restricción a los usuarios que utilizan la aplicación, podría ser distribuida en la Play Store ante un amplio marco de usuarios.

## 4.3. Presupuesto

Para la realización del presupuesto, se ha tenido en cuenta la planificación temporal realizada en apartados anteriores, que permite tener una visión del tiempo estimado que durará el proyecto, y, así, se ofrece la posibilidad de calcular el dinero necesario para cumplir con este proyecto en la duración estimada. Estos costes de los recursos y el personal se consideran costes directos, mientras que costes basados en elementos externos al proyecto (pago de la la luz, Internet, etc.) se consideran costes indirectos.

### 4.3.1. Coste del personal

Para empezar, se realiza el cálculo del presupuesto necesario para poder pagar a los miembros del proyecto un sueldo relativo a su experiencia. El equipo de desarrollo del proyecto está compuesto por dos miembros:

- **Luis Buceta Ojeda**. Investigador y desarrollador del proyecto, además de autor del presente documento.
- **David Griol Barres**. Tutor y director del proyecto.

En este caso, para calcular el presupuesto necesario para los costes de personal, no se tiene en cuenta la figura del tutor, que tiene un rol de orientador para el proyecto no reflejado en los costes de personal. En la Tabla 4.3, se especifica los costes de personal estimados utilizando una media aproximada de unos 20€/hora encontrados en el sueldo de un programador recién graduado.

| **Personal** | **Horas estimadas (a 13.5€/hora)** | **Total** |
|---|---|---|
| Luis Buceta Ojeda | 684 horas | 13680€ |

Tabla 4.3: Tabla de costes del personal final

### 4.3.2. Coste de los recursos

Por otro lado, también se ha calculado el precio de los recursos utilizados en el desarrollo del proyecto. Estos recursos se pueden dividir en dos variantes: costes de hardware y costes de software.

Para empezar, los costes de hardware engloban los dispositivos físicos utilizados para el desarrollo del sistema de simulación de entrevistas. En el caso de este proyecto, ha sido necesario utilizar un ordenador personal para poder utilizar las herramientas de desarrollo de Android, como Android Studio y todo el SDK de Android, y el acceso a las interfaces de los servicios externos que han permitido configurarlos para que funcionen correctamente integrados en el sistema de este proyecto. Este ordenador es un ordenador personalizado, de sobremesa y de propiedad del autor de este documento desde antes de comenzar el proyecto, cuyas características incluyen los siguientes componentes relevantes, descritos en la Tabla 4.4, junto con los precios de venta en el momento de su compra:

| **Componente** | **Nombre** | **Precio de venta recomendado (con IVA)** |
|---|---|---|
| Procesador (CPU) | Intel Core i7-8700K | 409€ |
| Tarjeta gráfica (GPU) | Geforce GTX1070 8GB | 519€ |
| Memoria RAM | Kingston DDR4 2400MHz, 16GB | 182€ |
| Placa base | Asus Z370-F | 196€ |
| Pantalla | Asus VS229HA | 109€ |
| **Total** | | **1415€** |

Tabla 4.4: Desglose de componentes del PC de desarrollo

Este ordenador, al ser un recurso que puede ser utilizado durante mucho tiempo, puede ser amortizado en cuatro años. Para calcular esta amortización, utilizamos la siguiente fórmula, donde A representa el tiempo de uso dentro del proyecto en meses, B la vida útil del recurso en meses y C el coste total del recurso en euros:

$$\frac{A}{B} * C = amortizacion \Rightarrow \frac{10}{48} * 1415 = \mathbf{294{,}79}$$

Además de los costes del ordenador, se ha tenido que utilizar un dispositivo Android para poder utilizar y probar la aplicación y así poder realizar el desarrollo. El dispositivo en cuestión ha sido un HTC One S, lanzado en 2012 y con un precio original de 379€, pero obtenido por 100€. Puesto que este terminal es antiguo, el periodo de amortización debe ser menor, de dos años como máximo. Utilizando la fórmula anterior, la amortización es la siguiente:

$$\frac{10}{24} * 100 = \mathbf{41{,}66}$$

Por último, destacar que los costes del software han sido 0€, ya que todas las herramientas de Android, los servicios externos utilizados y el software complementario (como GanttProject) son gratuitos.

### 4.3.3. Costes totales

El total de los costes del proyecto se pueden ver en la Tabla 4.5, teniendo en cuenta que los costes indirectos son un 20 % de los costes totales:

| **Componente** | **Nombre** |
|---|---|
| Costes de personal | 13680€ |
| Costes de hardware | 336,45€ |
| Costes de software | 0€ |
| Costes indirectos | 2803,29€ |
| **Total** | **16819,74€** |

Tabla 4.5: Presupuesto final del proyecto

Es necesario destacar que, tal y como se podrá ver en el siguiente apartado, el número de potenciales usuarios de la palicación es considerablemente alto, y es posible que los planes gratuitos de los servicios externos utilizados puedan no dar todos los recursos necesarios para que funcione correctamente el sistema ante una afluencia de usuarios que quieran utilizar la aplicación al mismo tiempo. Esto llevaría a la necesidad de mejorar el plan utilizado en cada uno de los cuatro servicios, ampliando los recursos disponibles a cambio de una tarifa mensual superior. Esto, potencialmente, podría aumentar los costes de software considerablemente, y es un elemento a tener en cuenta en el futuro del proyecto. Sin embargo, y como actualmente este proyecto comienza con un número de usuarios activos muy reducido, se mantienen los planes gratuitos de los cuatro servicios, ya que ofrecen los recursos suficientes para que el funcionamiento de la aplicación no se resienta, permitiendo que el presupuesto se mantenga congelado hasta la necesidad de utilizar estos planes de pago.

## 4.4. Entorno socioeconómico

En este apartado se realizará un análisis socioeconómico del entorno que rodea a la aplicación, y de los potenciales usuarios que podrían estar interesados en utilizar el sistema creado en este proyecto.

El primer punto a tratar es la definición del marco de potenciales usuarios que podrán utilizar el sistema. Este sistema de simulación de entrevistas está compuesto por un número de módulos que, en conjunto, es capaz de crear una interacción conversacional entre un usuario y un entrevistador virtual. Sin embargo, el usuario solo actúa directamente con uno de estos módulos: una aplicación disponible en la plataforma Android, que permite a los usuarios que utilicen sus dispositivos para utilizar el sistema. Android, tal y como se puede ver en el Estado del arte, es un sistema operativo disponible en un gran número de dispositivos móviles, y que no está cerrado a una sola marca de dispositivos como puede ocurrir en el sistema operativo iOS, que solo está disponible en dispositivos Apple. Los desarrolladores de Android, dada esta necesidad de adaptarse a una multitud de dispositivos, no puede ofrecer soporte para todos los dispositivos del mercado, dejando atrás los dispositivos más antiguos (que no reciben

actualizaciones para nuevas versiones de Android). En la Figura 2.7 y en la Tabla 2.1, podemos ver un desglose de la distribución de versiones de Android en los dispositivos activos en el mundo en 2018, y se puede ver que un 85 % de los usuarios utilizan una versión equivalente o superior a la versión 5.0 de Android (Lollipop). Dado este alto porcentaje de usuarios disponibles a partir de esta versión, la aplicación da soporte a cualquier dispositivo que tenga esta versión o superior. Teniendo en cuenta que hay más de 2000 millones de usuarios activos con Android en el mundo en 2017 [95], un 85 % de los usuarios implicaría que, potencialmente, esta aplicación tiene un número de usuarios potenciales superior a 1700 millones.

Por supuesto, este número no es lo suficientemente específico. El sistema de simulación de entrevistas, actualmente, solo funciona en castellano, por lo que el número de usuarios potenciales desciende considerablemente al estar limitado a usuarios hispanohablantes. Además, el sistema tiene una temática muy específica, y está destinado a la población activa que quieran practicar la realización de entrevistas de trabajo, por lo que está destinado a la utilización por parte de adultos que estén en el mercado laboral. De nuevo, tal y como se pudo ver en la introducción de este documento, en España hay más de 22 millones de ciudadanos que componen la población activa del país. Esto significa que, en España, hay más de 22 millones de potenciales usuarios que están interesados en realizar entrevistas de trabajo para o bien conseguir trabajo o bien conseguir puestos alternativos a los que tienen, y eso significa que pueden desear practicar este tipo de prácticas laborales de cara a futuras experiencias similares que puedan ayudarles en su carrera laboral. Además, España es un país donde, en 2017, cerca de un 40 % de los adultos menores de 25 años están en paro (y aquí se encuentran jóvenes con múltiples niveles de educación, habiendo personas con niveles de educación que abarcan desde secundaria hasta másteres) [96]. Las razones por este alto porcentaje de paro son varias, pero una de ellas es la falta de experiencia en los procesos de selección laborales, que si pueden tener otros candidatos con más experiencia. Es por ello que un sistema como el presentado en este Trabajo de Fin de Grado, puede ser de gran ayuda para los jóvenes que quieres comenzar a entrenar sus habilidades para saber responder a las preguntas de entrevistas laborales que pueden determinar el comienzo de su carrera laboral y que, potencialmente, ayudarían a reducir este porcentaje de paro entre adultos jóvenes.

Gracias a la utilización de una tienda integrada dentro del sistema Android, los usuarios tienen la posibilidad de obtener aplicaciones de Android, compatibles con sus dispositivos, de manera sencilla y rápida. Poder ofrecer a los usuarios de Android la capacidad de conseguir la aplicación de simulación de entrevistas en la Play Store permite que todos los usuarios de Android con una versión equivalente o superior a Android 5.0 y conexión a Internet puedan descargar a sus dispositivos la aplicación de este proyecto. Los desarrolladores, a la hora de distribuir una aplicación en la Play Store, puede decidir el precio que quiere pedir a los usuarios para que puedan obtener una licencia para descargar la aplicación, por lo que los desarrolladores pueden utilizar este sistema para conseguir beneficios con las aplicaciones distribuidas. Sin embargo, Google obtiene una comisión por cada venta realizada en la tienda, consiguiendo un 30 % del precio de la aplicación, dejando el 70 % restante al desarrollador [97], y el desarrollador está obligado a realizar un pago inicial a Google de 25€para poder dis-

tribuir su aplicación en la tienda. Por otro lado, los usuarios de Android tienen a no comprar aplicaciones en la tienda de Android, dado a que más del 95 % de las más de tres millones de aplicaciones disponibles [98] en la tienda son gratuitas [99], por lo que los usuarios tienen alternativas gratuitas a la gran mayoría de aplicaciones de pago encontradas en la Play Store. Es por ello, y también por el carácter educativo de la aplicación, que se ha determinado que, en caso de que la aplicación fuese distribuida en la tienda de Android, debería ser gratuita para todos los usuarios que quieran utilizarla, potencialmente utilizando el sistema de anuncios integrados ofrecido por Google (AdMob) [100] para conseguir beneficios por cada usuario que utilice la aplicación, pero de momento con el enfoque de ofrecer a los usuarios una experiencia educacional y que, así, puedan mejorar sus capacidades de respuesta en entrevistas de trabajo.

# Capítulo 5

# Conclusiones

Durante el desarrollo de este Trabajo de Fin de Grado, ha sido posible encontrar una multitud de conclusiones respectivas a las tecnologías utilizadas y al desarrollo de un proyecto compuesto por múltiples elementos que deben complementarse entre ellos, lo que implica el desarrollo en paralelo de una multitud de módulos que, al final, deben funcionar conjuntamente.

En los anteriores apartados, se ha podido observar como, de cara al entrenamiento de las habilidades necesarias para realizar una buena entrevista de trabajo, la población activa no tiene ninguna solución disponible para poder aprender como mejorar en estos aspectos, más allá de enfrentarse a varias de estas situaciones durante la longitud de su carrera laboral. Siendo esta la única solución disponible, los trabajadores más experimentados pueden, potencialmente, haber recogido estos conocimientos a lo largo de varios años y de haber vivido experiencias similares. Sin embargo, los adultos más jóvenes, que posiblemente acaben de terminar su educación, no tienen ningún tipo de experiencia en este ámbito, y esta falta de inexperiencia puede llegar a ser un detrimento importante en la que, muchas veces, suele ser la única prueba realizada en un proceso de selección laboral. Aunque, como se puede ver en el Estado del arte, hay un alto número de aplicaciones y recursos online que ofrecen a los usuarios consejos acerca de como abordar una situación así, hasta no estar en una entrevista real y recibir las preguntas del entrevistador no es posible preparar una respuesta que satisfaga la pregunta recibida. El principal objetivo de este Trabajo de Fin de Grado, además de los descritos en los primeros apartados, es realizar un estudio de si es posible utilizar la tecnología actual, que tan bien es aplicada en muchos ámbitos como apoyo, para ayudar a un amplio número de personas que, por un alto número de razones (o bien inexperiencia, o bien ansiedad social, o bien desconocimiento de que puede encontrar en un proceso similar), pueden no conocer la razón de su desacierto en los procesos laborales y quieren realizar un esfuerzo para mejorar ese aspecto de su vida, conociendo los conceptos en los que acertaría o fallaría si estuviese en una entrevista laboral real.

Este objetivo puede interpretarse de múltiples maneras, y este proyecto es solo una de las múltiples soluciones posibles para abarcar un problema que es considerablemente subjetivo, y que no puede ser interpretado objetivamente. Es por ello que, para este proyecto, se decidió utilizar una serie de análisis puramente subjetivos, como pueden ser los análisis de sentimientos o diccionarios de palabras claves, definidos exclusivamente

por un desarrollador con una opinión tan subjetiva como la de cualquier entrevistador. Este tipo de análisis subjetivo es la pieza clave de este sistema, y es el que permite separar este sistema de una grabadora de audio que quiere dar la impresión al usuario de que está realizando una entrevista real o una base de datos de consejos que, a pesar de ser útiles, no son aplicables para todas las situaciones. Con este análisis subjetivo, altamente ajustado para intentar ser lo más preciso posible respecto a las respuestas esperadas (tal y como se puede ver en la evaluación del sistema), el sistema es capaz de cumplir su objetivo funcional primordial, que es la capacidad de analizar las respuestas realizadas por un usuario ante una serie de preguntas realizadas por un entrevistador virtual, y poder mostrarle el resultado de estos análisis, para que pueda entender los errores o aciertos cometidos y así poder aplicarlos a entrevistas reales. La decisión de utilizar Android para la aplicación nexo también ha sido acertada, puesto que la posibilidad de utilizar Java (uno de los lenguajes de programación más utilizados en el mundo) permite que el desarrollo de la aplicación del usuario sea relativamente sencillo, permitiendo así que el desarrollo se haya podido centrar en otros aspectos, como el estudio de la solución, más que en la dificultad del desarrollo. Todo esto ha acabado sumando para lograr que el sistema logre cumplir su objetivo, con una precisión aceptable y resultados que podrían ser equivalentes a los encontrados por un entrevistador real planteando las mismas preguntas a un candidato, pero con la diferencia de que este sistema, al contrario que un entrevistador real, ofrece de manera transparente este análisis al usuario, y no se conforma con simplemente dar una respuesta positiva o negativa al candidato.

## 5.1. Trabajo futuro

La posibilidad de utilizar Android como núcleo del proyecto permite que una gran multitud de personas puedan utilizar este sistema usando una aplicación en los dispositivos móviles que utilizan diariamente, pero es cierto que gran parte de este sistema no está contenido dentro de esta aplicación de Android, ya que hace uso del desarrollo realizado en los servicios externos. Es por ello que uno de los objetivos futuros que podrían ser realizados en este proyecto es exportar este sistema a múltiples plataformas. Una aplicación en iOS sería un espejo a la arquitectura actual pero con la posibilidad de utilizar el sistema en dispositivos móviles Apple, pero también sería posible realizar una versión basada en tecnologías web, que funcionen en todos los dispositivos (móviles o de sobremesa) actuales. Aunque es cierto que con esto se abre el sistema a dispositivos que, potencialmente, pueden no tener el hardware necesario para interactuar correctamente con el sistema (por ejemplo, muchos PCs no tienen micrófonos disponibles para que el usuario pueda usarlos), gracias a que el sistema no está contenido dentro de la interfaz se pueden realizar ajustes a la transmisión de información entre los módulos del sistema para adaptarse a cada uno de los dispositivos en los que estuviese disponible.

Para la actual aplicación en Android hay también algunas mejoras disponibles que pueden ser interesantes en futuras iteraciones del proyecto, y que no han sido realizadas en esta primera iteración para centrar en progreso del proyecto en el estudio del sistema de simulación. Para empezar, sería posible utilizar el disco duro interno

de Android para guardar los resultados de previas entrevistas realizadas, y que así el usuario pudiese ver un historial de todas las simulaciones realizadas en su dispositivo. Además, uno de los elementos más interesantes que podrían añadirse es más servicios de análisis para las respuestas de los usuarios. Actualmente, solo se utiliza el texto como base del análisis, pero es posible encontrar servicios de análisis sonoro (que utiliza la onda del sonido para determinar elementos y sentimientos como enfado o duda, entre otros muchos, dentro de un archivo de sonido) y visual (que utilizarían la cámara del dispositivo para realizar un análisis gestual de la cara del usuario para determinar su emoción en el momento de realizar la respuesta) que podrían ayudar a que el análisis sea más preciso. Estos tipos de análisis fueron evaluados al comienzo del proyecto, pero por limitaciones del sistema Android, no es posible utilizar varios canales multimedia al mismo tiempo, por lo que no es actualmente posible realizar esta funcionalidad en Android, pero puede que en futuras versiones si se pueda implementar. Además de esto, por supuesto, la interfaz de la aplicación es mejorable, y quizá seguir un estándar como Material Design de Google ayude a que el diseño de la interfaz sea más agradable de utilizar para el usuario.

Por el lado de los servicios externos, también hay algunas potenciales mejoras que podrían ayudar a mejorar el sistema. En el lado de Dialogflow, siempre es posible ampliar el número de entidades y respuestas que el agente puede utilizar, para que los análisis sean más precisos. Además, podría ser interesante, de cara a futuro, cambiar la versión de la API utilizada a la más nueva para poder obtener todas las nuevas funcionalidades en el sistema, aunque eso implica esperar a que exista un SDK para Android que pueda utilizar esta nueva versión. Por el lado del *sentiment analysis*, quizá podrían implementarse otros servicios alternativos, que puedan ofrecer otra visión del sentimiento de la respuesta del usuario, o otros elementos de la minería de texto en general. Por último, tal y como se ha comentado anteriormente, en caso de aumentar el flujo de usuarios activos de la aplicación, sería necesario ampliar los planes de todos estos servicios, para obtener los recursos necesarios para ofrecer a todos los usuarios la misma experiencia.

En definitiva, y como última opinión subjetiva de este documento, este proyecto tiene todavía potencial para mejorar, tanto en funcionalidades como en experiencia de usuario, pero este Trabajo de Fin de Grado presenta los primeros pasos de una solución que puede ser de gran ayuda para un amplio marco de usuarios que no tienen un equivalente tecnológico al ofrecido por este sistema.

# Bibliografía


[1] M. Turk, «Multimodal interaction: A review», *Pattern Recognition Letters*, vol. 36, págs. 189-195, ene. de 2014, ISSN: 01678655. DOI: 10.1016/j.patrec.2013.07.003.

[2] C. Amma y F. Putze, «Multimodal Interaction, Design and Evaluation of Innovative User Interfaces», pág. 79,

[3] J. Bhaskar, K. Sruthi y P. Nedungadi, «Hybrid Approach for Emotion Classification of Audio Conversation Based on Text and Speech Mining», *Procedia Computer Science*, Proceedings of the International Conference on Information and Communication Technologies, ICICT 2014, 3-5 December 2014 at Bolgatty Palace & Island Resort, Kochi, India, vol. 46, págs. 635-643, 1 de ene. de 2015, ISSN: 1877-0509. DOI: 10.1016/j.procs.2015.02.112.

[4] (2018). Encuesta de Población Activa (EPA), dirección: http://www.ine.es/prensa/epa_tabla.htm.

[5] W. G. Kirkwood y S. M. Ralston, «Inviting Meaningful Applicant Performances in Employment Interviews», *The Journal of Business Communication (1973)*, vol. 36, n.º 1, págs. 55-76, 1 de ene. de 1999, ISSN: 0021-9436. DOI: 10.1177/002194369903600103.

[6] (2018). Dialogflow, dirección: https://dialogflow.com/.

[7] L.-F. Hurtado, J. Planells, E. Segarra y E. Sanchis, «Spoken dialog systems based on online generated stochastic finite-state transducers», *Speech Communication*, vol. 83, págs. 81-93, oct. de 2016, ISSN: 01676393. DOI: 10.1016/j.specom.2016.07.011.

[8] D. Griol, «Desarrollo y Evaluación de Diferentes Metodologías Para La Gestión Automática Del Diálogo», Universitat Politècnica de València, 2007.

[9] M. Johnston, S. Bangalore, G. Vasireddy, A. Stent, P. Ehlen, M. Walker, S. Whittaker y P. Maloor, «MATCH: An architecture for multimodal dialogue systems», Association for Computational Linguistics, 2001, pág. 376. DOI: 10.3115/1073083.1073146.

[10] B. Xiao, C. Girand y S. Oviatt, «Multimodal Integration Patterns in Children», pág. 5,

[11] A. B. Naumann, I. Wechsung y J. Hurtienne, «Multimodal Interaction: A Suitable Strategy for Including Older Users?», *Interacting with Computers*, Special Issue on Inclusion and Interaction: Designing Interaction for Inclusive Populations, vol. 22, n.º 6, págs. 465-474, 1 de nov. de 2010, ISSN: 0953-5438. DOI: 10.1016/j.intcom.2010.08.005.

[12] I. R. Titze, «Human Speech: A Restricted Use of the Mammalian Larynx», *Journal of Voice*, vol. 31, n.º 2, págs. 135-141, mar. de 2017, ISSN: 08921997. DOI: 10.1016/j.jvoice.2016.06.003.

[13] R. López-Cózar, J. Silovsky y M. Kroul, «Enhancement of emotion detection in spoken dialogue systems by combining several information sources», *Speech Communication*, vol. 53, n.º 9-10, págs. 1210-1228, nov. de 2011, ISSN: 01676393. DOI: 10.1016/j.specom.2011.01.006.

[14] E. Unal, K. Giakoumidakis, E. Khan y E. Patelarou, «Mobile phone text messaging for improving secondary prevention in cardiovascular diseases: A systematic review», *Heart & Lung*, mayo de 2018, ISSN: 01479563. DOI: 10.1016/j.hrtlng.2018.05.009.

[15] D. Hirafuji Neiva y C. Zanchettin, «Gesture Recognition: A Review Focusing on Sign Language in a Mobile Context», *Expert Systems with Applications*, vol. 103, págs. 159-183, 1 de ago. de 2018, ISSN: 0957-4174. DOI: 10.1016/j.eswa.2018.01.051.

[16] A. Krügel y R. Engbert, «On the Launch-Site Effect for Skipped Words during Reading», *Vision Research*, vol. 50, n.º 16, págs. 1532-1539, 21 de jul. de 2010, ISSN: 0042-6989. DOI: 10.1016/j.visres.2010.05.009.

[17] D. C. Derrick y G. S. Ligon, «The Affective Outcomes of Using Influence Tactics in Embodied Conversational Agents», *Computers in Human Behavior*, vol. 33, págs. 39-48, 1 de abr. de 2014, ISSN: 0747-5632. DOI: 10.1016/j.chb.2013.12.027.

[18] R. E. Guadagno, K. R. Swinth y J. Blascovich, «Social Evaluations of Embodied Agents and Avatars», *Computers in Human Behavior*, vol. 27, n.º 6, págs. 2380-2385, 1 de nov. de 2011, ISSN: 0747-5632. DOI: 10.1016/j.chb.2011.07.017.

[19] F. Manjoo, «A Murky Road Ahead for Android, Despite Market Dominance», *The New York TimesTechnology*, ISSN: 0362-4331.

[20] (2017). IDC: Smartphone OS Market Share, dirección: https://www.idc.com/promo/smartphone-market-share.

[21] (2018). Paneles de control, dirección: https://developer.android.com/about/dashboards/?hl=es-419.

[22] A. K. Jha y W. J. Lee, «An Empirical Study of Collaborative Model and Its Security Risk in Android», *Journal of Systems and Software*, vol. 137, págs. 550-562, 1 de mar. de 2018, ISSN: 0164-1212. DOI: 10.1016/j.jss.2017.07.042.

[23] (2018). SpeechRecognizer | Android Developers, dirección: https://developer.android.com/reference/android/speech/SpeechRecognizer.html.

[24] (2018). Adding Voice Capabilities | Android Developers, dirección: https://developer.android.com/training/wearables/apps/voice.html.

[25] (2018). Keyboard | Android Developers, dirección: https://developer.android.com/reference/android/inputmethodservice/Keyboard.

[26] (2018). Camera API | Android Developers, dirección: https://developer.android.com/guide/topics/media/camera.

[27] (2018). Android.hardware.camera2, dirección: https://developer.android.com/reference/android/hardware/camera2/package-summary.

[28] K. D'Silva. (2018). HandWave: A Library to Add Touch Free Capabilities to Mobile Applications, dirección: https://github.com/kritts/HandWave.

[29] (2018). Síntesis de voz de Google - Aplicaciones en Google Play, dirección: https://play.google.com/store/apps/details?id=com.google.android.tts&hl=es.

[30] (2018). TextToSpeech - Android Developers, dirección: https://developer.android.com/reference/android/speech/tts/TextToSpeech.

[31] (2018). OpenGL ES | Android Developers, dirección: https://developer.android.com/guide/topics/graphics/opengl.

[32] (2018). Vulkan Graphics API | Android NDK, dirección: https://developer.android.com/ndk/guides/graphics/?hl=es-419.

[33] (2018). Getting Started - OpenGL Wiki, dirección: https://www.khronos.org/opengl/wiki/Getting_Started.

[34] The Khronos Group. (7 de mar. de 2018). Khronos Group Releases Vulkan 1.1, dirección: https://www.khronos.org/news/press/khronos-group-releases-vulkan-1-1.

[35] (2018). Unity - Multiplatform, dirección: https://unity3d.com/es/unity/features/multiplatform.

[36] (2018). Unreal Engine Frequently Asked Questions, dirección: https://www.unrealengine.com/en-US/faq.

[37] (2018). Wiki - SAIBA - Mindmakers, dirección: http://www.mindmakers.org/projects/saiba/wiki.

[38] (2018). Dialogflow | Languages, dirección: https://dialogflow.com/docs/reference/language.

[39] S. Baraiya. (14 de oct. de 2017). Hello Jarvis!!! With DialogFlow, dirección: https://medium.com/indianic/hello-jarvis-with-dialogflow-5a038c2a3352.

[40] (2018). Dialogflow | Training, dirección: https://dialogflow.com/docs/training.

[41] (2018). Dialogflow | Entities, dirección: https://dialogflow.com/docs/entities.

[42] (2018). Dialogflow | Contexts, dirección: https://dialogflow.com/docs/contexts.

[43] (2018). Dialogflow | Intents, dirección: https://dialogflow.com/docs/intents.

[44] (2018). Dialogflow | Fulfillment, dirección: https://dialogflow.com/docs/fulfillment (visitado 31-05-2018).

[45] (2018). Dialogflow SDKs | Dialogflow, dirección: https://dialogflow.com/docs/sdks.

[46] (29 de mayo de 2018). Watson Assistant | Getting Started, dirección: https://console.bluemix.net/docs/services/conversation/getting-started.html#getting-started-tutorial (visitado 03-06-2018).

[47] R. High, «The Era of Cognitive Systems: An Inside Look at IBM Watson and How it Works», *IBM Corporation, Redbooks*, pág. 16, 2012.

[48] (11 de mayo de 2018). Watson Assistant | Supported Languages, dirección: https://console.bluemix.net/docs/services/conversation/lang-support.html#supported-languages.

[49] (2018). Watson Assistant - IBM Cloud, dirección: https://console.bluemix.net/catalog/services/watson-assistant-formerly-conversation.

[50] (26 de ene. de 2018). Watson Assistant | About, dirección: https://console.bluemix.net/docs/services/conversation/index.html#about.

[51] (30 de ene. de 2018). Watson Assistant | SDKs, dirección: https://console.bluemix.net/docs/services/watson/getting-started-sdks.html#sdks.

[52] B. Pang y L. Lee, «Opinion Mining and Sentiment Analysis», *Foundations and Trends® in Information Retrieval*, vol. 2, n.º 1–2, págs. 1-135, 7 de jul. de 2008, ISSN: 1554-0669, 1554-0677. DOI: 10.1561/1500000011.

[53] F. Poecze, C. Ebster y C. Strauss, «Social Media Metrics and Sentiment Analysis to Evaluate the Effectiveness of Social Media Posts», *Procedia Computer Science*, The 9th International Conference on Ambient Systems, Networks and Technologies (ANT 2018) / The 8th International Conference on Sustainable Energy Information Technology (SEIT-2018) / Affiliated Workshops, vol. 130, págs. 660-666, 1 de ene. de 2018, ISSN: 1877-0509. DOI: 10.1016/j.procs.2018.04.117.

[54] H. Tang, S. Tan y X. Cheng, «A Survey on Sentiment Detection of Reviews», *Expert Systems with Applications*, vol. 36, n.º 7, págs. 10 760-10 773, 1 de sep. de 2009, ISSN: 0957-4174. DOI: 10.1016/j.eswa.2009.02.063.

[55] N. Jakob, S. H. Weber, M. C. Müller e I. Gurevych, «Beyond the stars: Exploiting free-text user reviews to improve the accuracy of movie recommendations», ACM Press, 2009, pág. 57, ISBN: 978-1-60558-805-6. DOI: 10.1145/1651461.1651473.

[56] M. Ghiassi y S. Lee, «A Domain Transferable Lexicon Set for Twitter Sentiment Analysis Using a Supervised Machine Learning Approach», *Expert Systems with Applications*, vol. 106, págs. 197-216, 15 de sep. de 2018, ISSN: 0957-4174. DOI: 10.1016/j.eswa.2018.04.006.

[57] S. Murnion, W. J. Buchanan, A. Smales y G. Russell, «Machine Learning and Semantic Analysis of In-Game Chat for Cyberbullying», *Computers & Security*, vol. 76, págs. 197-213, 1 de jul. de 2018, ISSN: 0167-4048. DOI: 10.1016/j.cose.2018.02.016.

[58] A. Bermingham y A. F. Smeaton, «Classifying Sentiment in Microblogs: Is Brevity an Advantage?», en *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, ép. CIKM '10, New York, NY, USA: ACM, 2010, págs. 1833-1836, ISBN: 978-1-4503-0099-5. DOI: 10.1145/1871437.1871741.

[59] T. Yuz. (30 de abr. de 2018). A Sentiment Analysis Approach To Predicting Stock Returns, dirección: https://seekingalpha.com/article/4167585-sentiment-analysis-approach-predicting-stock-returns.

[60] V. Vyas y V. Uma, «An Extensive study of Sentiment Analysis tools and Binary Classification of tweets using Rapid Miner», *Procedia Computer Science*, vol. 125, págs. 329-335, 2018, ISSN: 18770509. DOI: 10.1016/j.procs.2017.12.044.

[61] (2018). MeaningCloud | Servicios Web de Analítica y Minería de Textos, dirección: https://www.meaningcloud.com/es.

[62] (2018). Upgrade Plan | MeaningCloud, dirección: https://www.meaningcloud.com/developer/account/subscriptions/upgrade.

[63] (2018). Integrations | MeaningCloud, dirección: https://www.meaningcloud.com/developer/integrations.

[64] (29 de mayo de 2018). Natural Language Understanding - IBM Cloud, dirección: https://console.bluemix.net/catalog/services/natural-language-understanding.

[65] (29 de mayo de 2018). Natural Language Understanding - Getting Started, dirección: https://console.bluemix.net/docs/services/natural-language-understanding/getting-started.html#getting-started-tutorial.

[66] (2018). Natural-language-understanding - API Reference | IBM Watson Developer Cloud, dirección: https://www.ibm.com/watson/developercloud/natural-language-understanding/api/v1/?python#introduction.

[67] (2018). AYLIEN | Text Analysis API, dirección: https://aylien.com/.

[68] (2018). Aylien - Pricing, dirección: https://developer.aylien.com/plans.

[69] E. Boiy y M.-F. Moens, «A machine learning approach to sentiment analysis in multilingual Web texts», *Information Retrieval*, vol. 12, n.º 5, págs. 526-558, oct. de 2009, ISSN: 1386-4564, 1573-7659. DOI: 10.1007/s10791-008-9070-z.

[70] (2018). El Proceso de Selección En Recursos Humanos - Wiki EOI de Documentación Docente, dirección: http://www.eoi.es/wiki/index.php/El_proceso_de_selecci%C3%B3n_en_Recursos_humanos.

[71] Massachusetts Institute of Technology. (2018). Interviewing Techniques and Structure, dirección: https://ist.mit.edu/sites/default/files/hr/interviewing_for_success/Interviewing%20Techniques%20and%20Structure.doc.

[72] Federación Valenciana de Empresarios de la Construcción. (2013). Estructura de una entrevista de selección, dirección: http://www.fevec.com/recursos_fevec/A1A7.pdf.

[73] D. Van De Mieroop y S. Schnurr, «Candidates' Humour and the Construction of Co-Membership in Job Interviews», *Language & Communication*, 9 de feb. de 2018, ISSN: 0271-5309. DOI: 10.1016/j.langcom.2018.01.002.

[74] (30 de mar. de 2018). iOS Developers Ship 29 % Fewer Apps In 2017, The First Ever Decline – And More Trends To Watch, dirección: https://blog.appfigures.com/ios-developers-ship-less-apps-for-first-time/.

[75] (2018). Interview Question and Answers - Aplicaciones en Google Play, dirección: https://play.google.com/store/apps/details?id=com.placecom.admobgoogleplayexample&hl=es.

[76] (2018). Mock Interview -Simulate REAL Interview Experience - Aplicaciones en Google Play, dirección: https://play.google.com/store/apps/details?id=com.placecom.texttospeech&hl=es.

[77] (2018). Job Interview Simulator - Aplicaciones en Google Play, dirección: https://play.google.com/store/apps/details?id=com.iqf.android&hl=es.

[78] C. L. Bovee y J. V. Thill, *Business Communication Essentials*, 7 edition. Boston: Pearson, 3 de ene. de 2015, 528 páginas, ISBN: 978-0-13-389678-7.

[79] C. L. Bovée y J. V. Thill, *Business Communication Today*, 13th edition. Boston: Pearson, 3 de ene. de 2015, 672 páginas, ISBN: 978-0-13-386755-8.

[80] (2018). Competencies - Practice job-level skills in VR - Aplicaciones en Google Play, dirección: https://play.google.com/store/apps/details?id=com.facetoface.vr&hl=es.

[81] (2018). Sentiment Analysis Request [2.1] | MeaningCloud, dirección: https://www.meaningcloud.com/developer/sentiment-analysis/doc/2.1/request.

[82] (2018). Sentiment Analysis Response [2.1] | MeaningCloud, dirección: https://www.meaningcloud.com/developer/sentiment-analysis/doc/2.1/response.

[83] (2018). Aylien | API Documentation, dirección: http://docs.aylien.com.

[84] Universidad Politécnica de Cartagena, Servicio Estudiantes y Extensión Universitaria, *Entrevista de Selección de Personal*, 2017.

[85] (2018). Top 50 Tough Interview Questions to Ask & Answer | Monster.com, dirección: https://hiring.monster.com/hr/hr-best-practices/small-business/conducting-an-interview/toughest-interview-questions.aspx.

[86] (2018). Dialogflow | /Query, dirección: https://dialogflow.com/docs/reference/agent/query.

[87] (2018). GanttProject: Free Desktop Project Management App, dirección: https://www.ganttproject.biz/.

[88] *Título I. De Los Derechos y Deberes Fundamentales - Constitución Española*, 6 de dic. de 1978.

[89] (14 de dic. de 1999). BOE.Es - Documento BOE-A-1999-23750, dirección: http://www.boe.es/buscar/doc.php?id=BOE-A-1999-23750.

[90] Agencia Española de Protección de Datos. (2018). Guía Del RGPD Para Responsables de Tratamiento, dirección: https://www.aepd.es/media/guias/guia-rgpd-para-responsables-de-tratamiento.pdf.

[91] (16 de nov. de 2017). Dialogflow - Standard Edition Terms of Service | Dialogflow, dirección: https://dialogflow.com/terms-and-privacy.

[92] (Mayo de 2018). MeaningCloud | Política de Protección de Datos, dirección: https://www.meaningcloud.com/es/politica-de-proteccion-de-datos.

[93] (2018). Privacy on the IBM Cloud | IBM Cloud, dirección: https://www.ibm.com/cloud/privacy.

[94] (2018). Aylien | Text API Terms of Service, dirección: https://aylien.com/legal/text-api-terms-of-service/.

[95] B. Popper. (17 de mayo de 2017). Google Announces over 2 Billion Monthly Active Devices on Android, dirección: https://www.theverge.com/2017/5/17/15654454/android-reaches-2-billion-monthly-active-users.

[96] (Abr. de 2018). Desempleo de España Desempleo menores de 25 2018, dirección: https://www.datosmacro.com/paro/espana?sc=LAB-25-.

[97] (2018). Comisiones de Transacciones - Ayuda de Play Console, dirección: https://support.google.com/googleplay/android-developer/answer/112622?hl=es.

[98] (2018). Number of Google Play Store apps 2018 | Statistic, dirección: https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/.

[99] (2018). Distribution of free and paid Android apps 2018 | Statistic, dirección: https://www.statista.com/statistics/266211/distribution-of-free-and-paid-android-apps/.

[100] (2018). Android Developers | AdMob, dirección: https://developer.android.com/distribute/best-practices/earn/show-ads-admob?hl=es-419.

# Apéndice A

# English summary

## A.1. Introduction

In this section, the current objectives and motivations behind this current Bachelor's degree final thesis will be defined.

### A.1.1. Motivation

The natural language processing (or NLU) is a field of study within the computer sciences (specifically, an element of the field of artificial intelligence) based on the interaction between the natural language used by humans every day and the language used by computer-based systems. The main objective of this science is to study and develop computer systems that allow their users to use the same language they use in their daily lives. Having this possibility, users can get closer to the system, acknowledging a familiar and recognizable experience, and being able to use past experiences to learn how to do the proposed activities by the system.

This is the main reason why natural language processing is one of the main foundations of multimodal interaction. Multimodal interaction is defined as the existence of multiple ways to interact with a system, allowing the users to use a variety of tools for information input and output (for example, audio, video or text, without limiting the system to only use one of these). With this kind of interaction, the system allows the users to have an adaptable experience using it, with better accessibility and usability. Being able to use multimodal systems in current devices can help the developers to gather up information from the users, and with the ability to use certain systems, like automatic speech recognizers, the user can interact with this systems with their own language and be recognized with the use of natural language processing techniques, and, potentially, use further analysis to extract more information from that input (using, for example, text mining techniques, which can extract information from any kind of text) and use this new information to assist users in the progress of the specific activities proposed by the system [2] [3].

One of the potential activities is known by more than 22 million Spanish citizens who make up the country's workforce [4]. Job interviews are formal meetings, between

a job opening candidate and an interviewer, with the objective of knowing the capabilities and attributes of this candidate in relation of a specific job opening. These interviews are essential to decide the new hiring the company wants to make, but for the candidate the result is a just a binary response (either the candidate gets hired or not). Interviewers don't usually offer any kind of feedback of the candidate's performance in the interview, so they're unable to fully understand what had gone right or wrong. Knowing this, and the previously mentioned technologies, a job interview simulator with feedback functionalities could help a sizeable amount of users to improve in these kind of situations, letting them know what their virtues and failures are in an interview [5].

### A.1.2. Objectives

For this thesis, the main objective is to create a job interview simulation system, that, using multimodal interaction based on speech and text, is able to offer to the users feedback of their performance in these simulations.

The user will interact with a conversational agent that will have two roles:

- It will keep the flux of the conversation, making questions to the users, receiving their answers.
- It will study, using semantics, the user's answer, also analysing the relation between this answer and the proposed question.

On the other hand, the agent will rely on a parallel analysis system, based on text mining and textual sentiment analysis, which will study the tone and sentiment of the user's response, to understand the intention of this response. This analysis, along with the one performed by the conversational agent, will be weighted by the system and shown to the user as an evaluation of his or her participation in the simulation, so that the user can interpret its results properly and improve when answering certain questions in real job interviews.

Finally, although the basis of this system could be defined in multiple platforms that could perform the same functions, this project defines the system within an Android environment, an operating system present in most of the mobile devices currently available.

## A.2. State of the Art

The main objective of this chapter is to detail the theoretical and technological context in which this project is found.

### A.2.1. Spoken dialog systems

A spoken dialogue system is a computer system that receives, from a user, an input based on phrases made with natural and oral language, and returns an output with phrases generated, again, with natural and oral language [7]. On an abstract level, this definition is also applicable to an oral conversation between two humans, because a spoken dialog system seek that the interaction between a user and a computer system is as close as possible to the one with another natural person.

The basic structure of a spoken dialog system is defined with the following modules [8]:

- **Automatic speech recognition system**, also known as **Speech-To-Text (STT)**. This system recognizes the sounds emitted by the user's speech, and extracts the information contained in this speech to transform it into text.
- **Spoken language comprehension**. This module studies the received text from the previous module and performs a semantic analysis of it.
- **Dialog manager**. Using the comprehension analysis, this module is in charge of generating the new text response that will be transmitted to the user in the last module.
- **Information query module**. This module performs queries to external services, such as databases or external information servers.
- **Response generator**, also known as **Text-To-Speech (TTS)**. It's in charge of sending to the user the final response generated by the dialog manager, synthesizing the text to voice.

Spoken dialog systems can be planned with a multimodal approach. Using multimodal interaction, users get more efficient at completing the tasks proposed by the system, thanks to the possibility of using multiple interaction channels and being able to use channels where the user is more comfortable (both in abilities and accessibility). For a spoken dialog system, having the possibility of using multimodal interaction allows the users to use not only text input, but also alternative input methods like his or her own speech. Being able to use their own language and speech creates a familiar environment in the users, and makes that the information received is more natural and truthful than a text-based input [1]. With current technology, both the input and the output of almost every system can become multimodal. The input, in general, uses text, speech or gesture-based entries to communicate with the system, while the output, normally, uses the device's hardware to present the information in text, speech or using 3D graphics (using, for example, virtual human representations to show the information to the user, like a real person would) [13].

In the case of this project, the user will interact with a spoken dialog system using an Android application. Android is an operating system based on Unix, available on a multitude of mobile devices like smartphones, smartwatches or TVs. It's used by 85 % of the smartphones users of the world, on top of the 14.7 % of iOS (Apple's own operating system) and the 0.1 % of Windows Phone (from Microsoft) [20]. Given the big amount of mobile devices Android is in, Android suffers from a fragmentation problem: device manufacturers must approve individually each Android update, and, in most cases, old devices are left without any kind of support for new versions of Android. Because of that, any application created for Android should be planned with the knowledge that more than a 63 % of the users use an Android version older than the last two releases (7.0 Nougat and 8.0 Oreo), and that not all functionalities of these last versions can be applied yet to the general audience [22]. Android, being in mobile devices with multiple hardware input systems, can be used to create a multimodal system like the one introduced in this thesis, and Android has integrated tools (like Speech-To-Text, Text-To-Speech, virtual keyboards and graphics rendering) that can be used by developers to create a multimodal system that can be used in most devices.

To conclude, there are several online services for creating spoken dialog systems, such as Dialogflow [6] and Watson Assistant [49], that allows the defining of conversational agents that are able to recreate part of the functionality of a spoken dialog system using technologies like machine learning and natural language processing to create a conversation with a user.

### A.2.2. Sentiment analysis

Sentiment analysis is a branch of text mining (a process based on the retrieval of information within a text using technologies like machine learning or statistics), in charge of studying fragments of text to recognize the subjective tone and intention which was used to create it. This tone, usually defined as a positive, negative or neutral value called polarity, represents a key concept in the identification of the context in which a text (written using natural language) has been formed. It's mainly used to obtain a picture of the creator's intention when he or she was creating the text, and that can be used to understand the perception of a specific subject by this creator. One of the many uses of this science is that, using sentiment analysis, various companies use the customers' reviews to understand the rating of certain products offered, and the perception of the quality of each of the qualities of those products by the users, which then can be used to improve the products themselves and obtain better ratings and better customer satisfaction [53] [55].

For doing sentiment analysis, statistical models must be used, and in recent years machine learning techniques have been implemented to improve these models automatically, adapting the model's patterns and dictionaries to improve the analysis accuracy. Also, because of how the analysis algorithm divides the text into smaller fragments with syntactic logic and studying them first separately to then join them again together, the sentiment analysis algorithms are able to recognize long texts without problem, but the analysis is more accurate in smaller texts that don't add any kind of noise to the input [58]. The process to prepare and analyse any kind of text involve

seven steps that any kind of sentiment analysis system must go through: text input (where the text enters the system through a variety of channels), tokenization (where the original text gets divided into subdivisions with syntactic logic called tokens), stop word filtering (where these tokens get filtered to remove unnecessary words, like articles or pronouns, that don't add anything to the analysis), negation handling (where the system recognizes the negation expressions and adjust the analysis to better understand the meaning of the text), stemming (where the system gets to the root, or stems, of every token, to understand the relationships between each of them), classification (where, knowing all the relations between the stems, it starts analysing each of these separately, rebuilding the original text step by step) and sentiment class (where, having all the previous classifications, it calculates the polarity of each element and the whole text) [59].

To conclude, and just like with spoken dialog systems, there are multiple services online that offer the possibility to integrate sentiment analysis solutions into any system with connection to Internet. Some of these services are MeaningCloud [61], Watson Natural Language Understanding (Watson NLU) [65] or Aylien [67], which can offer a variety of text mining-related functionalities, like sentiment analysis, to developers, using their free or paid APIs.

### A.2.3. Recruitment processes

A job recruitment process is a procedure, carried out by one or more companies, with the purpose of finding a series of candidates for a job, carrying out a study of several of their skills in relation to that job, and proceeding, through several phases, to filter candidates until the candidate who best suits the job offered is found. This process is made up of several phases necessary for the Human Resources (HR) department to carry out the necessary filters and the necessary checks to obtain a full analysis of all the candidates to then choose the perfect candidate for the job [70].

While there are multiple phases that a candidate can go through during a recruitment process, the main phase, and the only one that it's obligatory, is a face-to-face job interview. In this phase, the candidate and one or more interviewers get in a formal meeting where the interviewers ask questions relative to the experience of the candidate (about education achievements and past jobs, for example), and the candidate must answer them. The interviewers do these questions with the intention of learning new facts about the candidate that may be relevant to the job they're offering. This, for the candidate, is usually the last step in the recruitment process, and in some cases, it is the only step. Given that a candidate may not have other opportunities to get hired in that process, this is a critical phase for the candidate, and it's not possible for him or her to make a rehearsal of it before it happens (mainly because every single interview is different, and its contents depend on what the interviewer is looking for in a candidate). This fluid structure of an interview prevent that candidates can be fully ready for it and, especially for young and unexperienced candidates, the lack of knowledge about how to answer to the interview's questions can be a big detrimental in the interviewer's opinion of the candidate, and can end up with a negative rating of the company towards him or her [73].

### A.2.4. Similar applications

This project uses an Android application as the main entry point for the users to interact with the system. Android allows users to download new applications from the integrated application store (Google Play Store) to any compatible devices they own. Just in 2017, developers released more than one and a half million applications to the store [74], and, because of that, the store is full of applications about a big multitude of activities and functionalities. Knowing this fact, it's not farfetched to believe that in the store it's possible to find applications that try to help users with job interviews. In the following list, some of the applications found in the Play Store get analysed and compared with the one proposed in this document:

- **Interview Questions and Answers** (free, more than a million downloads) [75]: it has a database of questions and quizzes that can be found in a job interview, but the user can only interact with it to read these resources (no job interview simulation of any kind).
- **Mock Interview** (free, more than 50000 users) [76]: complement of the previous application, it allows users to face a fake interview, but the interaction to the system is limited to record the responses. There's no analysis made for these responses, so the user doesn't get any kind of rating or feedback of the interview they made.
- **Interview Simulator** (free with extra fees for more resources, more than 5000 downloads) [77]: this one allows to do simulations of job interviews via audio or video (multimodal), and by the end it offers a complete evaluation of the user's performance. In testing, the results obtained weren't precise enough, especially in the relevance section of the results, so it doesn't seem like a reliable tool for teaching users.
- **Competencies** (free version very limited, more than 5000 users) [80]: this VR application allows users to enter a virtual 3D room, where they're simulating a job interview. It has user's performance ratings, like eye contact made by the user virtual avatar, but the chosen parameters to rate this performance are not relevant for a real job interview in most cases.

In conclusion, Interview Simulator offers the closest experience to the one developed in this thesis, but there are elements of it that can be polished, such as the recognition of text information, which can be achieved with other tools as seen in previous State of the art sections.

## A.3. Design and Development

In this section, the phases of the design and development of the application will be defined in a summarized manner.

### A.3.1. System definition

This project, being an Android application, requires critical functionalities that users can use. The application is based on the idea of simulating job interviews, where the user plays the role of the interviewee, and the application modules do the interviewer work. It is essential that the application performs this simulation mainly using voice. Voice is one of the key elements of human communication, allowing other people to express both information and emotions, and they can interpret this data and generate a response. In a simulation of this style, where two people participate, the communication is carried out directly between each participant, and both react in sequence to each other's comments. In the context of an interview, it is an extremely weighted conversation, where the interviewee responds with the detail he or she considers necessary to the questions that the interviewer asks. Therefore, the application must simulate only one of the participants: the interviewer. The system must be able to recognize the user's voice (using a Speech-To-Text module) and collect the information it contains, read this information, evaluate it and respond to the user with other questions that continue the interview. In addition, it must allow multiple ways of interaction, users must be able to interact with both speech and typing responses. Offering both possibilities, users can interact through the most suitable channel for them, both in terms of accessibility and comfort. On the other hand, the role of the interviewer must be represented through output channels and use these channels to transmit information back to the user. That is why, through the application, the system will communicate with the user using both text and synthesized voice (thanks to a Text-To-Speech module), making this output a mirror of the user's input. In addition to asking the user new questions, the system must be able to perform an analysis of the user's answers. The system will use technologies based on natural language processing, such as keyword recognition and text mining, to conduct a study of the input proposed by the user, and thus achieve an assessment of the user's performance in answering the questions posed. Finally, this evaluation should be shown to the user, so that he or she can have a reference of his or her performance and then be able to improve in future simulations.

The main functionality of this project is the following: to **simulate a face-to-face job interview**, where the candidate will be the user and the interviewer will be the system represented by the application itself. Although the user will interact with an Android application, this application will only be the user's interface with the analysis system located below, which will be composed of a conversational agent based on natural language processing and sentiment analysis systems. With this defined, this project has three differentiated key elements: a conversational agent, a group of feeling analysis systems and an application that encapsulates the interaction with these systems.

The **conversational agent** (designed using Dialogflow, a machine learning-based conversational service) must be able to complete the following objectives:

- **Model customization**. The developer of the model must be able to adjust the internal model of the conversational agent. This model is composed, mainly, by intents and entities, which are the main elements that are needed to be defined to make the agent fully aware of the conversation it's having with the user. The intents are defined by the training phrases needed to recognize any intent, and the responses that the agent will return in case of finding itself within a certain intent. Entities, on the other hand, are dictionaries of keywords that can be defined by the developer and used in the entities to understand certain elements of the input. This model will be defined to allow multiple interviews, but the questions will be defined in a specific order that cannot be changed, to allow the agent to understand the order of the answers made by the user.
- **User's intent recognition**. The agent must be able to recognize the meaning of the information received as input from the user, and it should identify to which intent of its internal model it relates to.
- **Entities recognition**. Once the intention of the input is recognized, the agent must access its dictionary of entities and recognize the potential keywords found within the information received. These entities define the relevance of the answer to the question asked.
- **Question generation**. The conversational agent must know which question is the next one that it's going to ask to the user, because these can be formulated in several ways, depending on the model's customization.
- **Sending results to the application**. Finally, the conversational agent will send all the information retrieved in the previous phases (intents, entities and the new question to be asked to the user) to the Android application for it to manage it locally. This information can be sent in several formats, in this case it will arrive as a JSON that can be accesed with Dialogflow's SDK.

On the other hand, the **group of sentiment analysis services** (MeaningCloud, Watson NLU and Aylien, each one independently from the rest) must be able to achieve the following goals:

- **Global polarity generation**. Sentiment analysis services should be able, given an input of plain text, to recognize the feeling and tone of the text as a whole, and not just the polarity of each of its terms. They may have extra capabilities, such as subjectivity analysis, language elements recognition like irony or sarcasm, or a ratio of confidence, that can complement the analysis and, therefore, obtain more accurate results that can be used in the Android application. In this case, all three are able to give the polarity of the full text plus the confidence of the given result, and in the case of MeaningCloud, it's possible to customize the internal analysis model (not the case in the other two).

- **Sending results to the application**. The services must be able to communicate the results obtained back to the application, containing, in a JSON format, all the parameters discovered in the analysis (mainly, the overall polarity of the text, but also the optional elements they may find in their analysis). In this case, Watson NLU can use an SDK for Android to receive this information, but MeaningCloud and Aylien require of standard HTTP requests to communicate with the system.

Finally, the **Android application** (designed with Android 5.0 as minimum required version, and developed using Android Studio, the official development enviroment of Android) will be able to complete the following functionalities:

- **Show questions to the user**. Since the beginning of an interview, the application must show questions, received from the conversational agent, to the user, using both the screen to show it in text form and the audio hardware of the devices, used to play back a synthetic voice (generated using the Text-To-Speech modules integrated in Android) that dictates the current question.

- **Manage the input of the user's answers**. The application must allow the users to input the answers they want to introduce into the system using two main ways: speech (using the Speech-To-Text module integrated in the Android SDK, that transforms the voice of the user to text that can be used by the other modules of the system) and text (using virtual or physical keyboards to write the answers).

- **Connect with the analysis systems**. The application must connect with the external analysis systems: the conversational agent (which will send the new proposed question, the number of keywords found in the previous answer and a mark to signal the end of the interview) and the group of sentiment analysis services (which will send back the polarity of the previous answer, as calculated by each one of them). Each of these connections will be made using either the SDKs of each of the services or HTTP requests, and each one will happen in a different execution thread (to use the multithread capabilities of modern devices).

- **Results evaluation**. Once the application receives all the analysis results from the external services, it will use an internal algorithm to join all of these results into one final result that will represent the user's performance answering the current question. This algorithm calculates first the sentiment analysis results, using a weighted average to comprehend the results of MeaningCloud, Watson NLU and Aylien, and then it calculates the result of the keyword search in the conversational agent. Having these two analysis, it calculates the final result of the user (in a scale of 0 to 1, where 0 is the most negative result and 1 the most positive) in this particular question.

- **Final report**. When the application reaches the end of the interview, it generates a report of the performance of the user, using all the results calculated in the previous step. This report has a global final score (from 0 to 100 %, where less than 50 % is a failure) and a breakdown of all the questions and answers of the

interview, with all the results achieved by the user (divided in the results achieved by the sentiment analysis services and the ones achieved by the conversational agent).

### A.3.2. Testing and results

Since much of the system's functionality depends on the accuracy of the results given by the analysis algorithm found in the Android application's interview module, it is necessary to conduct a specific study of the performance of this algorithm, to obtain an adequate weighting of each of the individual results of the external sentiment analysis services integrated into the system. This algorithm is a weighted average between the polarities and the confidences of the results obtained in the sentiment analysis carried out by MeaningCloud, Watson NLU and Aylien, which ends up resulting in a final polarity that represents the final result of the user when answering a question. In the following Table A.1, the results of the initial testing of each of the sentiment analysis services used, given multiple types of responses, are specified:

| **Answer** | **MeaningCloud** | **WatsonNLU** | **Aylien** |
|---|---|---|---|
| Positive | 0.789 | 0.814 | 0.792 |
| Positive (double negation) | 0.632 | 0.733 | 0.788 |
| Negative | 0.484 | 0.259 | 0.311 |
| Negative (double negation) | 0.436 | 0.321 | 0.298 |
| Doubtful | 0.412 | 0.520 | 0.502 |
| Doubtful (double negation) | 0.381 | 0.495 | 0.513 |

Tabla A.1: Sentiment analysis results given multiple response tones

Each of the sentiment analysis services gets different results, but one is different from the other two. MeaningCloud can correctly recognize the simplest sentences (without any double denials). However, as soon as the sentence becomes more complicated and begins to use more complex structures, such as the double negation defined earlier that usually gives trouble to text mining services, MeaningCloud stops giving the expected results and begins to have problems with the polarity of the response. It's possible to see, for example, that in the two types of positive responses studied (positive responses with or without double negations), simple sentences recognize them positively without any problem, but by introducing double negation the polarity is reduced in a bigger way than in the other two services (which, in comparison, maintain a similar polarity, especially Aylien who is particularly successful in recognizing double negations). On the other hand, the doubt is correctly studied in the three services, but MeaningCloud obtains a more negative polarity than the rest of the services, which is increased even more when more complex linguistic structures are introduced and is not consistent with the analysis carried out by Watson NLU and Aylien, which are more precise in their assessments. Watson NLU appears to have a larger range of results than the

other two, although equally accurate, while Aylien appears to be the most stable of the three services.

Seeing these results, it was decided that, given the unpredictability of MeaningCloud in some analyses, its result cannot be weighted the same as the results of Watson NLU and Aylien, because both offer more precision and stability than the first one. Therefore, the algorithm weighs MeaningCloud slightly lower (with a 0.2 weight in the average), while Watson NLU and Aylien, given the similar results they offer, are weighted equally (with a 0.4 weight each). In the following Table A.2, the results of having this weighting is shown:

| **Response** | **Resultado** |
|---|---|
| Positive | 0.799 |
| Positive (double negation) | 0.739 |
| Negative | 0.353 |
| Negative (double negation) | 0.335 |
| Doubtful | 0.495 |
| Doubtful (double negation) | 0.485 |

Tabla A.2: Results of the weighted algorithm, using multiple types of responses

As shown, the values are more accurate with this new weighting (high values for positive comments, low values for negative comments, relatively neutral but somewhat negative values for doubtful comments and the double negation more consistent with simpler sentences), and the instability caused by MeaningCloud has been reduced thanks to the use of a higher weighting with the other two services, which give more precise results.

On top of this, other tests have been made to the system, specifically to the application, which is the main interaction point for the user, and all of them passed. Given that the interviews can be resolved and analysed in multiple ways, and each user can make the responses he or she sees fit, the system cannot be tested with multiple users, because each one would expect something different that the results they might show. Because of that, the tests needed to be defined for just the developer, who knows what should be expected in each situation.

## A.4. Project management

In this section, the legal context of this system and the socio-economic environment of the potential users who will use it will be defined.

### A.4.1. Regulatory framework

The first concept studied in this legal context is a study of the legislation adhered to a project of this nature. In a job interview, the interviewer asks the candidate about his or her past experience in education and other work environments, and, given the intention to carry out a realistic simulation of this process, this system of simulations of interviews defined in this project raises similar questions to the user, so the answers received by the system can potentially be private. This means that the application must be governed by several legal fronts, starting with Spanish law. In Spain, the Spanish Constitution marks the protection of this kind of private data as a constitutional right of all citizens of the country, in Article 18.4 [88]:

> "The law shall limit the use of data processing in order to guarantee the honour and personal and family privacy of citizens and the full exercise of their rights."

This article is supported by a recent law, which is mandatory in every country in the European Union, called GDPR (or General Data Protection Regulation, approved in 2016 and implemented on 25th May 2018). This European law prevents any company that works in Europe to store any kind of personal and private information about their users without explicit consent. In this project, the application doesn't store any information locally, but the developed system contacts with multiple external services, so it's necessary to check that every single one of these services comply with GDPR. Luckily, the three sentiment analysis services (MeaningCloud, Watson NLU and Aylien) [92] [93] [94] and the conversational agent (Dialogflow) [91] comply with GDPR, so the users that use this system can be assured that their info is not stored anywhere (and because the system doesn't ever ask the name to user, the information sent is always sent without any context of who has used the system). This compliance with GDPR allows the developer to distribute the application in the Google Play Store, that requires that the application follows the laws of the region it launches in.

### A.4.2. Socio-economic enviroment

In this section, a socioeconomic analysis of the environment surrounding the application and of the potential users who might be interested in using the system created in this project will be performed.

The first point to be discussed is the definition of the potential users who will be able to use the system. This interview simulation system is composed of several modules that together can create a conversational interaction between a user and a virtual interviewer. However, the user only acts directly with one of these modules: an application available on the Android platform, which allows users to use their devices

to use the system. 85 % of users use a version equivalent or higher than Android 5.0 (Lollipop) [21]. Given this high percentage of users available from this version onwards, the application supports any device with this version or higher. Considering that there are more than 2 billion active users with Android in the world in 2017 [95], 85 % of users would imply that this application potentially has a number of potential users in excess of 1.7 billion. In addition, Spain is a country where, by 2017, about 40 % of adults under the age of 25 were unemployed [96]. That is why a system such as the one presented in this thesis can be of great help to young people who wants to begin the training of their skills to know how to answer the questions of job interviews that can determine the beginning of their working career and that, potentially, would help to reduce this percentage of unemployment among young adults.

Thanks to the use of the shop integrated into the Android system, users have the possibility of obtaining Android applications, compatible with their devices, easily and quickly. Being able offer Android users the ability to get the interview simulation application through the Play Store allows all Android users with a version equivalent or superior to Android 5.0 and Internet connection to download to their devices the application of this project. Developers, when distributing an application on the Play Store, can decide the price they want to ask users for the application, which can be used to get benefits with distributed applications. Due to the educational nature of the application, it has been determined that, in the event that the application is distributed in the Android store, it should be free of charge for all users who want to use it, potentially using the integrated advertising system offered by Google (AdMob) [100] to get benefits for each user who uses the application, but, for the time being, with the focus on offering users a free educational experience, which can improve their capabilities in job interviews.

## A.5. Conclusions

In the previous sections, it has been detailed how, to train the necessary skills to carry out a good job interview, the active population does not have any available solution to learn how to improve in these aspects, beyond facing several of these situations during the length of their working career. As this is the only solution available, younger adults, who may have just finished their education, do not have any experience in this field, and this lack of inexperience can be a major handicap in what is often the only test of a recruitment process. Although, as you can see from the State of the Art, there are many online applications and resources that offer, to their users, tips on how to deal with this kind of situations, it is not possible to prepare an answer that satisfies the question received until you are in a real interview and receive the interviewer's questions. The main objective of this Bachelor's degree final thesis, in addition to those described in the first sections, is to conduct a study of whether it is possible to use current technology to help a large number of people who, for a number of reasons (like inexperience, social anxiety, or lack of knowledge of what you may encounter in a similar process), may not know the reason for their failure in recruitment processes and may want to make an effort to improve this specific aspect of their life, by knowing the concepts they would have gotten right or wrong in a real job interview.

This objective can be interpreted in multiple ways, and this project is just one of many possible solutions to address a problem that is largely subjective and cannot be interpreted objectively. Using subjective analysis (the sentiment analysis and the analysis realized by the conversational agent, which both simulate the thought process made by a real interviewer), the system can fulfil its primary functional objective, which is the ability to analyse the answers made by a user to a series of questions asked by a virtual interviewer, and to show him or her the result of these analyses, so that he or she can understand the errors or successes made and then apply this knowledge to real interviews. The decision to use Android for the nexus application has also been the right one, since the possibility of using Java (one of the most widely used programming languages in the world) helped the development of the user's application to become relatively simple, allowing the developer to focus on other aspects, such as the definition of the solution. All of this has helped to make the system achieve its objective, with good accuracy and results that could be equivalent to those found by a real interviewer asking the same questions to a candidate, but with the difference that this system, unlike a real interviewer, offers this analysis to the user in a transparent way, and does not limit itself to just giving a positive or negative answer to the candidate.

### A.5.1. Future work

There are many future developments that could improve the system, and one of these is to export this system to multiple platforms. An application on iOS would be a mirror to the current architecture on Apple mobile devices, but it would also be possible to make a version based on web technologies, which work on all current devices (mobile or desktop). Although it is true that this opens the system to devices that potentially may not have the necessary hardware to interact properly with the

system (for example, desktop PCs without microphones), thanks to the fact that much of the system is not contained in the application, adjustments can be made to the transmission of information between the system modules to adapt to each of the devices used.

For the current Android application there are also some improvements available that may be of interest in future iterations of the project, that couldn't be made in this first iteration to focus the project on the study of the simulation system. To begin with, it would be possible to use Android's internal hard drive to save the results of previous interviews, so that the user could see a history of all the simulations performed on their device. In addition, more analysis services for user responses could be added (like audio or video analysis). In addition to this, of course, the application's interface can be improved, and perhaps following a standard like Google's Material Design could help in making the interface more user-friendly.

On the external services side, there are also some potential improvements that could help to improve the system. About Dialogflow, it is always possible to expand the number of entities and responses that the agent can use, so that the analyses are more accurate. Also, if the number of active users of the application were to increase, it would be necessary to expand the resources for all these services to bring good quality of service to all the users (upgrading to paid plans).

Finally, and as the last subjective opinion of this document, this project still has potential to improve, both in functionalities and in user experience, but this thesis presents the first steps of a solution that can be of great help for a wide range of users that may want to use the technology to improve their skills in this particular subject.